Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Diplomová práce

2020 **Bc. Luboš Beran**

Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Využití služeb Národní Identitní Autority v rámci přijímacích řízení vysokých škol

Diplomová práce

Bc. Luboš Beran

Školitel: Ing. Marta Vohnoutová

Konzultant: Ing. Luděk Rašek

České Budějovice 2020

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL MAGISTERSKÉ PRÁCE

Student: Bc. Luboš Beran
(jméno, příjmení, tituly)

Obor – zaměření studia: Informační systémy a technologie

Katedra/ústav PřF, kde proběhne obhajoba: Ústav aplikované informatiky

Školitel: Ing. Marta Vohnoutová
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: Ing. Luděk Rašek
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma magisterské práce: Využití služeb Národní Identitní Autority v rámci přijímacích řízení vysokých škol

Cíle práce :

Teoretická část:

- Identifikace zákonných pověření pro možné použití Národní identitní autority v rámci vysokých škol jako správního orgánu pro přijímací řízení.
- Provést věcný rozbor ukotvení elektronické identity v právním řádu ČR a EU zejména v souvislosti s oprávněním organizace využít eIDAS elektronickou identitu.
- Popis existujících softwarových modulů pro implementaci služeb Národní identitní autority,

Praktická část:

- Návrh řešení implementace služeb Národní identitní autority pomocí známých softwarových modulů
- Vytvoření proveditelností studie nasazení implementace v rámci informačního systému vysoké školy.
- Zdrojový kód SW modulů je povinnou součástí práce

Základní doporučená literatura:

http://www.szrcr.cz/uploads/eIdentita/SeP_pr_i_ruc_ka_SZR_0v11_2018_02_06.pdf

https://info.eidentita.cz/zakony/2502017/ a související předpisy EU (eIDAS a prováděcí předpisy)

https://doi.org/10.6028/NIST.SP.800-63-3 a související

http://openid.net/connect/

https://wiki.oasis-open.org/security/FrontPage

Financování práce .....

Školitel práce: Ing. Marta Vohnoutová podpis : .....

U externích vedoucích fakultní garant práce ..... podpis : .....

Garant oboru mag. studia doc. RNDr. Iva Dostálková, Ph.D. podpis : .....

Vedoucí katedry/ústavu, kde bude práce vypracovávána: Ing. Rudolf Vohnout, Ph.D.

podpis : .....

Případný souhlas vedoucího ústavu AV ..... podpis : .....

V Českých Budějovicích dne 25.10.2019 podpis studenta .....

Bibliografické údaje

Beran L., 2020: Využití služeb Národní Identitní Autority v rámci přijímacích řízení vysokých škol [Use of services Narodni Identitni Autorita within the admission procedures of universities Mgr. Thesis, in Czech] - 121 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Diplomová práce se zabývá vypracováním systému, který umožní využívat služeb Národní identitní autority v rámci přijímacího řízení na vysokých školách. Využití by měl, ale najít i v jiných veřejných institucích. V teoretické části jsou vysvětleny pojmy související s Národní identitní autoritou a jejím fungováním v rámci EU. Práce řeší i to, zda jsou vysoké školy oprávněné využívat služeb Národní identitní autority. V návrhové části je prezentována plánovaná architektura a vybraný software včetně jeho popisu. Implementační část se věnuje instalaci a konfiguraci softwaru tak, aby práce působila jako návod pro sestavení funkčního celku pro využití služeb Národní identitní autority.

Klíčová slova

Národní identitní autorita, eIDAS, SAML, OpenID Connect, Keycloak, Apache Tomcat, Spring Boot

Annotation

This master's thesis deals with the development of a system that will be able to use the services of the Narodni identitni autorita in the framework of the admission procedure at universities. It can be also used in other public institutions. The theoretical part explains terms related to Narodni identitni autorita and its functioning within the EU. The thesis also deals with whether universities are entitled to use the services of the Narodni identitni autorita. The architectural part presents the planned architecture and selected software. The implementation part deals with the installation and the configuration of software so that the thesis can be used as a guide for building a functional unit for the use of the services of the National Identity Authority.

Key words

Narodni identitni autorita, eIDAS, SAML, OpenID Connect, Keycloak, Apache Tomcat, Spring Boot

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne _____ _____

podpis

Poděkování

Chtěl bych tímto poděkovat paní Ing. Vohnoutové, že se mě ujala a nabídla mi zpracování zajímavé diplomové práce. Dále bych rád poděkoval panu Ing. Raškovi za jeho pomoc, rady a trpělivost odpovídat na moje zvídavé otázky.

Obsah

Seznam pojmů

Informační systém základních registrů – systém veřejné správy, který sdílí data mezi základními registry a agendovými informačními systémy.

Nařízení eIDAS – Nařízení Evropského parlamentu a Rady (EU) č. 910/2014, *o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu* [1].

Elektronická identifikace – postup používání osobních identifikačních údajů v elektronické podobě, které jednoznačně a nepopiratelně určí fyzickou či právnickou osobu [1; 2].

WebAuthn – standard webové autentizace, který využívá asymetrické kryptografie.

eGovernment – digitální a moderní veřejná správa.

SAML – standard založený na jazyce XML pro popis a výměnu autentizačních a autorizačních dat [3].

Autentizace – elektronický postup, který jednoznačně umožňuje potvrdit elektronickou identifikaci fyzické či právnické osoby nebo původ a integritu dat v elektronické podobě [1].

Interoperabilita – schopnost dvou a více systémů nebo jejich částí poskytovat si informace a služby a efektivně spolupracovat [4].

Prováděcí akt – právně závazný akt, který pod dohledem výborů, jež se skládají ze zástupců členských států EU, umožňuje stanovit podmínky v zájmu jednotného uplatňování právních předpisů EU [5].

Autorizace – přidělení oprávnění, kterými může disponovat přihlášená osoba.

Single Sign-On – uživatel se může přihlašovat k webům a aplikacím jedné společnosti.

OpenID Connect – **autentizační** protokol založený na OAuth 2.0, který definuje atributy o uživateli a specifikuje endpointy. Protokol podporuje federovanou autentizaci.

OAuth 2.0 – **autorizační** protokol, který podporuje delegovanou autorizaci.

Federovaná autentizace – uživatel se může přihlašovat k webům a aplikacím různých společností s použitím jedné identity.

Delegovaná autorizace – typ autorizace, který podporuje granulární oprávnění. Aplikace vyžadující oprávnění dostává od autorizačního serveru pouze práva nezbytná k provedení akce.

Seznam zkratek

ISDS – Informační systém datových schránek

OVM – Orgán veřejné moci

ISZR – Informační systém základních registrů

SZR – Správa základních registrů

eOP – Elektronický občanský průkaz

DESI – The Digital Economy and Society Index

EGDI – E-Government Development Index

NIA – Národní identitní autorita

IdP – Identity Provider

SeP – Service Provider

SONIA – Soukromoprávní NIA

DS – Datová schránka

AIFO – Agendový identifikátor fyzické osoby

ZIFO – Zdrojový identifikátor fyzické osoby

SePP – Service Provider Pseudonym

IdPP – Identity Provider Pseudonym

LoA – Level of Assurance

SAML – Security Assertion Markup Language

SSO – Single Sign-On

RÚIAN – Registr územní identifikace, adres a nemovitostí

VPS – Virtuální privátní server

IAM – Identity and Access Management

JSON – JavaScript Object Notation

OIDC – OpenID Connect

JWT – JSON Web Token

RP – Relying Party

OP – OpenID Provider

1 Úvod

O digitalizaci státní správy České republiky se mluví již několik let. V roce 2007 byl spuštěn projekt *Czech POINT*, kde na terminále bylo možné zažádat o výpisy z obchodního a živnostenského rejstříku a z katastru nemovitostí. V roce 2009 zahájil provoz *Informační systém datových schránek* (ISDS), který slouží k výměně dokumentů mezi orgány veřejné moci (OVM) a fyzickými a právnickými osobami. Digitalizaci státu podpořil vznik *Informačního systému základních registrů* (ISZR), který řídí *Správa základních registrů* (SZR). Zrodem ISZR byly propojeny, do té doby nejednotné a nespolupracující, základní registry. Další významný milník přišel v roce 2012, kdy byly vydány první **elektronické občanské průkazy** – eOP (občanské průkazy s kontaktním čipem). Průkazy se ovšem v té době setkaly s nedostatečnou oporou v českém právu, se špatným technickým řešením a občanům nepřinášel elektronický čip téměř žádné využití ani v běžném životě, ani v komunikaci se státní správou. Významným krokem ze strany EU bylo schválení Nařízení Evropského parlamentu a Rady č. 910/2014, *o elektronické identifikaci a službách vytvářející důvěru pro elektronické transakce na vnitřním trhu* (**Nařízení eIDAS**). V souladu s nařízením eIDAS byly přijaty v roce 2018 nové eOP, a to již s funkcí **elektronické identifikace**. eOP je možné použít při přihlašování k online státním službám a také při podepisování kvalifikovaným elektronickým podpisem. V blízké době čeká eOP další evoluce, protože Nařízení Evropského parlamentu a Rady č. 2019/1157, *o posílení zabezpečení průkazů totožnosti*, zavádí povinnost bezkontaktního čipu [6]. To je příležitost zavést autentizační standard WebAuthn, ale vše záleží na implementaci ze strany Ministerstva vnitra, protože nařízení WebAuthn explicitně nezmiňuje [7].

Dalším krokem ze strany EU ke zpřístupnění služeb občanům členských států, který se týká také vysokých škol, je projekt *Jednotné digitální brány* (Nařízení Evropského parlamentu a Rady č. 2018/1724, *kterým se zřizuje jednotná digitální brána pro poskytování přístupu k informacím, postupům a k asistenčním službám a službám pro řešení problémů*). Cílem je poskytnout ucelený státní rozcestník k nejdůležitějším informacím pro občany a podnikatele. Vysokých škol se nařízení dotýká zejména tím, že je čeká vytvoření jednotného přístupového rozhraní např. pro přihlášky ke studiu nebo pro uznání diplomu. V souvislosti s tímto

nařízením uvolnilo Ministerstvo školství, mládeže a tělovýchovy 170 miliónů korun na digitalizaci vysokých škol [8].

Cílem celého snažení digitalizace státu je mít moderní digitální veřejnou správu – eGovernment. Konkrétním vrcholným cílem eGovernmentu ČR je, aby platilo, že

„*Česká republika je jednou z předních zemí v praktickém využívání moderních služeb eGovernmentu, což významně přispívá k přívětivosti a celkové efektivitě výkonu veřejné moci*“ [9].

Jako zhodnocení úspěchu plnění tohoto cíle je pozice ČR v žebříčku dle *DESI* indexu Evropské komise. V žebříčku roku 2020 se ČR nacházela na 17. místě z celkových 28 míst [10]. Oproti žebříčku roku 2018 je vidět zlepšení o 2 pozice. ČR je nadprůměrná v oblasti Integrace digitálních technologií. Naopak ztrácí v oblastech Konektivita a Digitální veřejné služby.

S digitalizací státní správy je svázán program *Digitální Česko*, který je definován pojmem „Strategie koordinované a komplexní digitalizace České republiky 2018+“ [11]. Navazující strategií na tento program je *Inovační strategie České republiky 2019-2030* [12]. Cílem strategie je posunout ČR mezi nejinovativnější země Evropy. O lepší vztah mezi státními službami a občany usiluje i *Iniciativa 202020*. Ta se snaží o to, aby ČR byla do roku 2020 v hodnocení eGovernmentu, podle *EGDI* žebříčku, sestavovaným Organizací spojených národů, mezi 20 nejlepšími zeměmi v Evropě. Podle tohoto žebříčku vydaného v roce 2018 se ČR nachází na 25. místě z 28 zemí Evropské unie, které mají eGovernment [13].

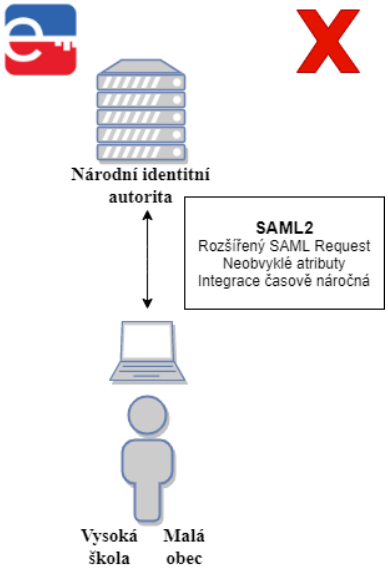
Z výše uvedených informací, lze spatřovat snahu o digitalizaci ČR. Úsilím této diplomové práce je **propojit Národní identitní autoritu**, hlavní bod eGovernmentu, **s informačním systémem** vysoké školy **a zajistit převod získaných atributů** do modernějšího formátu. Práce je rozdělena na teoretickou, návrhovou a implementační část. V **teoretické části** je popsána Národní identitní autorita a komunikace mezi domácími a zahraničními službami. Následuje úsek, který se věnuje vysokým školám a tomu, zda mají možnost vystupovat jako kvalifikovaný poskytovatel online služeb. **Návrhová část** se zabývá návrhem architektury. Součástí je popis SAML zpráv, které Národní identitní autorita používá. Poté je vybrán modernější autentizační protokol a softwarové komponenty. V poslední části diplomové práce již probíhá **implementace vybraných prvků** do použitelného celku, který bude schopen

komunikovat s Národní identitní autoritou a předávat atributy do informačního systému vysoké školy. **Hlavním přínosem práce je převod uživatelských atributů z jazyka XML do modernějšího formátu a také usnadnění připojení k Národní identitní autoritě**. Výsledkem práce je použitelný **postup implementace** nejen **pro vysoké školy**, ale i další **veřejnoprávní subjekty**, jako jsou **městské a obecní úřady**. Základním předpokladem pro dosažení cíle je využít open source projekty.

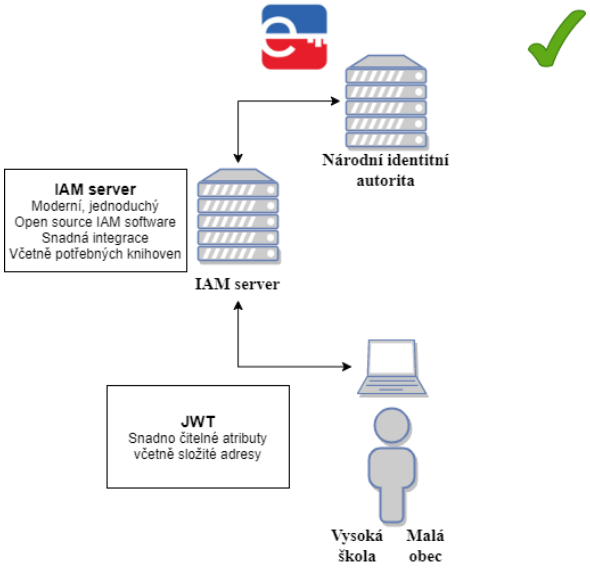
1.1 Cíle práce

- Popsat Národní identitní autoritu a s ní spojené pojmy.
- Seznámit se zákony týkající se Národní identitní autority.
- Seznámit se zákony týkající se vysokých škol jako orgány veřejné moci.
- Provést věcný rozbor ukotvení elektronické identity v právním řádu České republiky a Evropské unie.
- Popis existujících softwarových modulů pro implementaci služeb Národní identitní autority.
- Vytvoření studie proveditelnosti implementace služeb Národní identitní autority do informačního systému vysoké školy.

Pro lepší vizualizaci praktických cílů práce jsou přiloženy ilustrační obrázky.



Obrázek 1 - Současné řešení



Obrázek 2 - Požadované řešení

Diplomová práce řeší především zjednodušení integrace služeb Národní identitní autority pro vysoké školy a menší orgány veřejné moci. Zajištěn je i převod zasílaných nestandardních atributů do srozumitelného podoby. **Výstupem je řešení, které dovolí orgánům veřejné moci přistoupit ke službám Národní identitní autority snadno, rychle a bez složitého nastavování**.

2 Národní identitní autorita

Národní identitní autorita (**NIA**, také známá jako Národní bod pro identifikaci a autentizaci) zprostředkovává služby důvěryhodných poskytovatelů identit – **Identity Provider** (IdP), jednotlivým důvěryhodným poskytovatelům služeb – **Service Provider** (SeP), kteří vyžadují důvěryhodnost autentizací přistupujících uživatelů prostřednictvím kvalifikovaného systému. NIA udržuje vazbu mezi základní elektronickou identitou fyzické osoby a instancemi elektronické identity této osoby u SeP. Správcem NIA je SZR [14].

NIA je celek, federativní systém, který se skládá z několika komponent [2]:

- kvalifikovaného správce (IdP),
- národního bodu,
- národního uzlu eIDAS,
- základních registrů.

Na seznamu IdP, kteří svými prostředky poskytují identifikaci a autentizaci fyzické osoby, jsou v době psaní práce 4 služby (uvedeny v kapitole 2.1.1 Kvalifikovaný systém a kvalifikovaný správce). V roce 2021 se seznam rozšíří o bankovní identitu. O zařazení mezi IdP usilovala i služba mojeID od sdružení CZ.NIC a dne 13.8.2020 získala akreditaci.

V současnosti se na seznamu SeP nachází přes 50 využitelných služeb. Jsou zde zastoupeny organizace jak státní správy (uvedeny v kapitole 2.1.2 Kvalifikovaný poskytovatel online služeb), tak soukromá společnost Sazka a.s. V průběhu psaní práce se v seznamu nenachází vysoká škola.

Novelou zákona č. 21/1992 Sb., *o bankách*, vzniká systém (SOukromoprávní NIA – SONIA), který umožní službám využívat přihlášení bankovní identitou [14]. Od 1.1.2021 by mělo být možné přihlásit se k elektronickým službám soukromoprávního i veřejnoprávního sektoru bankovní identitou. Již nyní jsou vidět snahy o propojení bank a NIA v testovacím prostředí.

2.1 Zákonná ustanovení

2.1.1 Kvalifikovaný systém a kvalifikovaný správce

Kvalifikovaný systém je takový systém elektronické identifikace, který splňuje zákonem stanovené podmínky. Systém splňuje specifikace, normy a postupy pro alespoň jednu úroveň záruky podle Prováděcího nařízení Komise 2015/1502 [15]. Kvalifikovaný systém musí poskytovat služby NIA. Jeho spravování má na starosti kvalifikovaný správce systému elektronické identifikace (IdP) [16].

Povinností kvalifikovaného správce je vydat **prostředek pro elektronickou identifikaci** v souladu s prováděcím nařízením výše. Jako prostředek pro elektronickou identifikaci se rozumí **hmotná** (eOP) či **nehmotná** (jméno, heslo, SMS) **jednotka obsahující identifikační údaje**, která se používá k **autentizaci** pro účely online služby [1]. To, zda vydaný prostředek splňuje či nesplňuje požadavky stanovené zákonem, určuje osoba pověřená ministerstvem [16].

Mezi prostředky poskytující důvěryhodnou službu identifikace a autentizace se řadí v ČR:

- eOP (vydávaný Ministerstvem vnitra ČR),
- mobilní klíč eGovernmentu (vydávaný Ministerstvem vnitra ČR),
- jméno, heslo a SMS (vydávané SZR),
- čipová karta Starcos (vydávaná I.CA),
- účet mojeID (sdružení CZ.NIC, z.s.p.o.),
- mezinárodní prostředky pro elektronickou identifikaci (mezinárodní uzel eIDAS).

Kvalifikovaný správce je **pouze státní orgán** nebo **osoba**, které je udělena akreditace pro správu kvalifikovaného systému. Akreditaci je možné získat podle §5 zákona č. 250/2017 Sb., *o elektronické identifikaci* [16].

Zákonem další stanovené povinnosti kvalifikovaného správce jsou zejména **zajištění dostupnosti** jím spravovaného kvalifikovaného systému způsobem umožňujícím dálkový

přístup prostřednictvím národního bodu a pro národní bod způsobem umožňujícím dálkový přístup [16].

Správce má také za úkol **ověřit totožnost** držitele před prvním použitím prostředku pro elektronickou identifikaci prostřednictvím národního bodu a **zapsat identifikátor** prostředku pro elektronickou identifikaci a jeho **úroveň záruky do národního bodu** [16]. Národní bod nabízí ověření držitele skrze napojené registry a je nutné uvést, jakými prostředky pro elektronickou identifikaci držitel disponuje.

Správce je povinen vést evidenci jím **vydaných prostředků** pro elektronickou identifikaci. Tato evidence je soubor jak obecných osobních údajů, tak specifických záznamů o prostředku pro elektronickou identifikaci [16]. Mezi obecné osobní údaje patří:

- jméno,
- příjmení,
- adresa trvalého pobytu,
- datum narození držitele.

Mezi specifické záznamy se řadí:

- identifikátor prostředku pro elektronickou identifikaci,
- identifikátor držitele v rámci kvalifikovaného systému,
- údaj o způsobu ověření totožnosti žadatele o vydání prostředku pro elektronickou identifikaci,
- datum a čas vydání prostředku pro elektronickou identifikaci,
- doba platnosti prostředku pro elektronickou identifikaci.

2.1.2 Kvalifikovaný poskytovatel online služeb

Kvalifikovaným poskytovatelem online služeb (SeP) se rozumí **poskytovatel online služeb** nebo jiných činností, **který umožňuje prokázání totožnosti s využitím elektronické identifikace** [1; 17]. Takové prokázání totožnosti by měly vyžadovat právní předpisy nebo výkon působnosti poskytovatele. Poskytovatel je povinen o této skutečnosti neprodleně

informovat správce národního bodu a uvést jakou službu poskytuje a jakou úroveň záruky prostředku pro elektronickou identifikaci požaduje [16]. Zákon o elektronické identifikaci dále neupravuje požadavky na poskytovatele.

OVM, které chtějí nabídnout své služby skrze NIA a využívat bezpečného a zaručeného ověření totožnosti uživatelů, se musí registrovat jako kvalifikovaný poskytovatel online služeb. Toho je možné dosáhnout na Portále národního bodu. Po přihlášení do zákonem zřízené datové schránky (DS) následuje konfigurace kvalifikovaného poskytovatele. V konfiguraci se specifikuje:

- IČO subjektu,
- název kvalifikovaného poskytovatele,
- popis kvalifikovaného poskytovatele,
- URL adresa s informacemi o kvalifikovaném poskytovateli,
- unikátní URL adresa zabezpečené části webu,
- adresa pro příjem vydaného tokenu,
- URL adresa, na kterou bude uživatel přesměrován po odhlášení,
- adresa pro načtení veřejné části šifrovacího certifikátu z metadat NEBO
    - je nezbytné importovat šifrovací certifikát z lokálního disku,
- logo kvalifikovaného poskytovatele.

Do stále rostoucí skupiny online služeb poskytovaných OVM patří např.:

- Portál občana,
- eRecept,
- ePortál ČSSZ,
- Český báňský úřad,
- Klientský portál Ministerstva práce a sociálních věcí.

Soukromoprávní subjekty, kterým zákon dává povinnost ověřovat totožnost a věk, mohou využít ověření totožnosti skrz kvalifikovaný systém elektronické identifikace [18]. Těmto

subjektům není umožněno přímé registrace jako OVM, ale musí zaslat datovou zprávu do DS SZR [19]. Příkladem takových subjektů jsou provozovatelé hazardních her, kterým zákon č. 186/2016 Sb., *o hazardních hrách*, ukládá povinnost ověření totožnosti zákazníka [18].

2.1.3 Národní bod

Národní bod je zřízen zákonem č. 250/2017 Sb., *o elektronické identifikaci* [16]. Zákon definuje národní bod jako **informační systém veřejné správy**, který podporuje **proces elektronické identifikace a autentizace prostřednictvím kvalifikovaného systému**. Součástí národní bodu je **eIDAS uzel**, čímž se rozumí místo připojení, které je součástí architektury Interoperabilita elektronické identifikace [20]. Nařízením eIDAS je tak zajištěná možnost předávání informací o identifikaci a autentizaci osob mezi jednotlivými členskými státy EU.

V národním bodu jsou uvedeny tyto záznamy [16]:

- identifikátor prostředku pro elektronickou identifikaci a jeho úroveň záruky,
- agendový identifikátor (AIFO) držitele pro agendu autentizace,
- identifikátor držitele v rámci kvalifikovaného systému.

AIFO i identifikátor držitele jsou vygenerovány v informačním systému ORG. Systém ORG byl vyvinut pro nahrazení používání rodného čísla v základních registrech bezvýznamovými identifikátory. ORG je jediným místem, kde jsou k dispozici všechny identifikátory. Mělo by tak být dosaženo vyšší ochrany osobních údajů [21].

AIFO slouží pro identifikaci konkrétní osoby v rámci agendy [22]. Není možné z AIFO odvodit zdrojový identifikátor a nelze z něj ani dovodit osobní nebo jiné údaje o fyzické osobě. AIFO vychází ze zdrojového identifikátoru fyzické osoby (ZIFO) a kódu agendy. V každé agendě má uživatel nový AIFO. Ze ZIFO není možné odvodit osobní ani jiné údaje o fyzické osobě, jíž byl přiřazen [22].

Pro komunikaci mezi národním bodem, SeP a IdP jsou používány **bezvýznamové směrové identifikátory**, které jsou nazývány jako **pseudonymy**. Jsou to:

- Service Provider Pseudonym (SePP),

- Identity Provider Pseudonym (IdPP).

Z pseudonymů není možné dovodit osobní ani jiné údaje. Zároveň SeP neví, jakého IdP uživatel využil pro identifikaci a autentizaci. To platí i obráceně. IdP neví, k jaké službě se osoba přihlašuje.

Národní bod je napojen k základním registrům, které umožňují vyzvednutí požadovaných informací o osobě a předání SeP. Využívané registry tvoří [16]:

- základní registr obyvatel,
- informační systém evidence obyvatel,
- informační systém cizinců,
- základní registr agend, orgánů veřejné moci, soukromoprávních uživatelů údajů a některých práv a povinností.

2.2 Úroveň záruky

Úroveň záruky (Level of Assurance – **LoA**) je definovaná v nařízení eIDAS jako **míra spolehlivosti prostředků** pro elektronickou identifikaci při určování totožnosti osob [1]. Jde o míru jistoty, která určuje, že ten, kdo se vydává za určitou identitu, je skutečně osoba spojená s touto identitou.

Pro vyjádření LoA se přihlíží k [1]:

- postupům (prokazování a ověřování totožnosti a autentizace),
- řídícím činnostem (subjekt vydávající prostředky pro elektronickou identifikaci a postup vydávání těchto prostředků),
- prováděným technickým kontrolám.

Jednotlivé LoA se dělí na úrovně [1; 17]:

- nízkou – low,
- značnou – substantional,
- vysokou – high.

Jednotlivé stupně se řídí podle toho, jak silné je **ověření osoby** a jaký je **používán prostředek** pro elektronickou identifikaci:

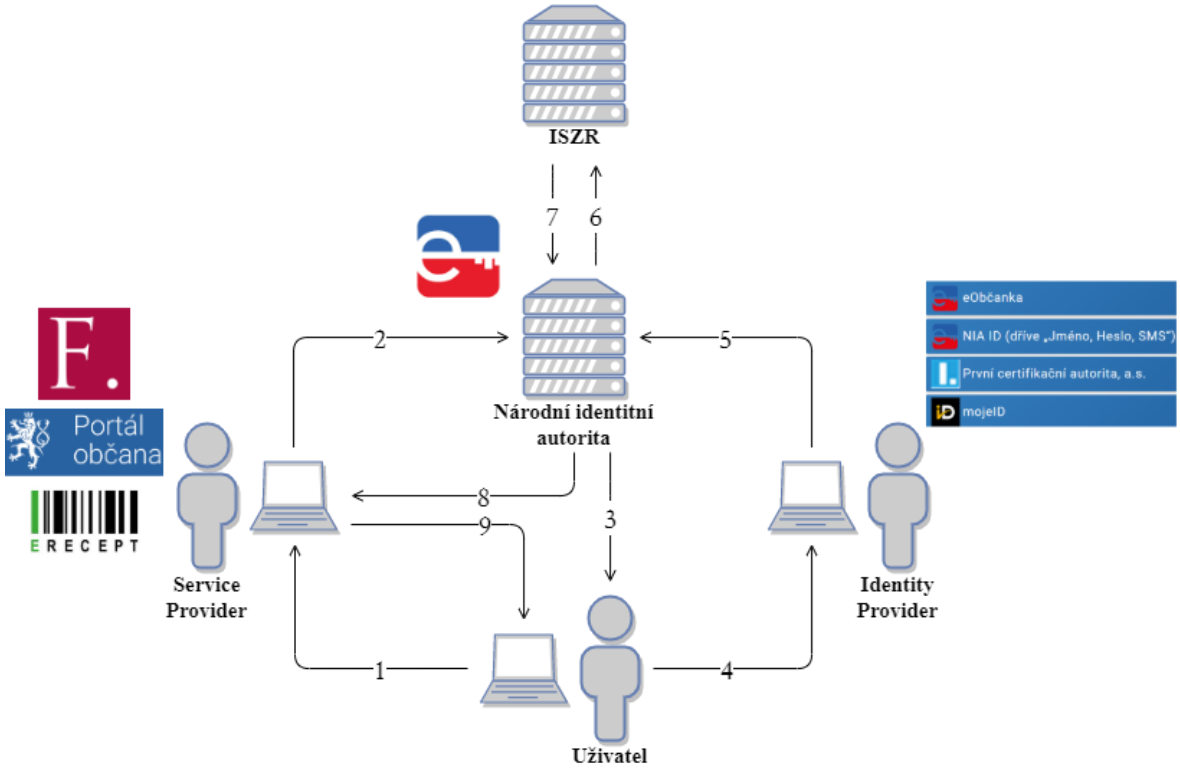
- LoA vysoká požaduje fyzický prostředek pro elektronickou identifikaci na bezpečném zařízení, kdy při předání byla zaručeně ověřena uživatele totožnost a že zná přístupová hesla.
    - Přihlášení pomocí eOP.
- LoA značná se deklaruje dvoufaktorovou autentizací a ověřením totožnosti žadatele.
    - Přihlášení jménem, heslem a SMS.
- LoA nízká nepožaduje zaručené ověření totožnosti.
    - Přihlášení jménem a heslem u ISDS.

Národní bod upraví nabídku IdP podle toho, jakou LoA SeP potřebuje.

- Specifikovaná přesná (exact) LoA např. značná → dostupná LoA je pouze značná.
- Specifikovaná nejnižší (minimum) požadovaná LoA např. značná → dostupné LoA jsou značná a vysoká.
- Pokud není uvedena konkrétní LoA, jsou nabídnuty všechny dostupné.

2.3 Proces ověření uživatele pro využití online služeb

Celý proces ověření uživatele ilustruje následující Obrázek 3 - Ověření uživatele



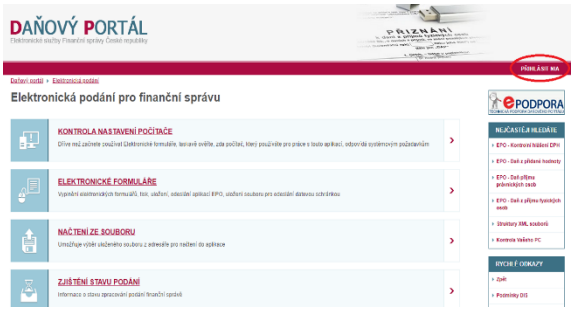
Obrázek 3 - Ověření uživatele

Krok číslo	Popis
1	Uživatel vyžaduje službu od SeP. SeP potřebuje důvěryhodnou informaci o tom, kdo chce využít jeho služeb.
2	SeP připraví, pomocí standardu SAML, žádost o identifikaci, kterou zašle NIA. V žádosti definuje: • požadovanou LoA, • seznam požadovaných atributů o uživateli. Uživatel je přesměrován na vstupní bod NIA.
3	NIA uživateli nabídne seznam dostupných IdP, které splňují požadavky ze žádosti.
4	Uživatel se ověří vůči zvolenému IdP.
5	Pokud je uživatel ověřen, NIA je zaslán autentizační token s IdPP.
6	NIA žádá ISZR o vydání atributů ze žádosti.
7	ISZR vydává požadované atributy. Vždy je vyzvednut údaj kontrolující úmrtí osoby. Pokud osoba zemřela, je celá transakce zrušena.
8	NIA posílá SeP požadované atributy včetně SePP.
9	Uživatel je identifikován a autentizován a SeP uživateli poskytuje žádanou službu.

Tabulka 1 - Proces ověřování uživatele

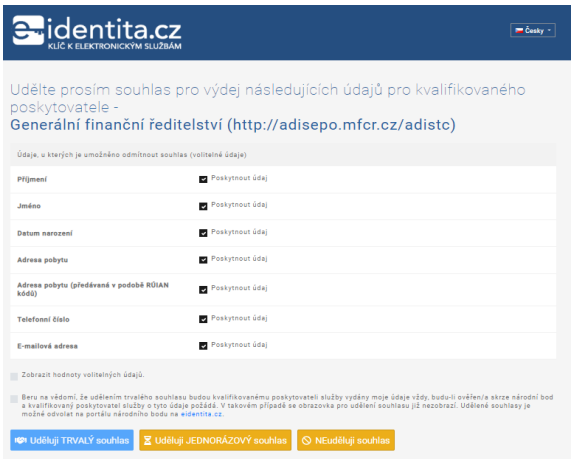
2.3.1 Přímý a nepřímý model elektronické identifikace

Výše uvedený proces je nazýván jako **nepřímý model**. Následující obrázky nepřímý model ilustrují. Pro ukázku je zvolena služba Daňový portál.



Obrázek 4 – Nepřímý model – Portál se službou

Obrázek 5 – Nepřímý model – Volba IdP



Obrázek 6 – Nepřímý model – Souhlas s údaji

Obrázek 7 – Nepřímý model – Zpřístupnění služby

Obrázek číslo	Popis
3	Uživatel vyžaduje službu od SeP.

4	Uživatel je přesměrován na vstupní bod NIA, kde jsou mu zobrazeny dostupní IdP. Požadovány jsou:
	• LoA alespoň nízká,
	• uživatelské atributy:
	  ○ jméno,
	  ○ příjmení,
	  ○ datum narození,
	  ○ adresa pobytu,

	o telefonní číslo, o emailová adresa, o pseudonym.
5	Uživatel se vůči IdP ověří a je vybídnut k odsouhlasení výdeje údajů.
6	Pokud vše proběhlo správně, uživateli se zobrazí předané údaje a zpřístupní se mu požadované služby (např. Kontrolní hlášení DPH nebo Přiznání k dani z příjmů).

Tabulka 2 - Nepřímý model

Roli prostředníka zajišťuje NIA [23]. NIA dbá o důvěryhodnost transakce. Výhody jsou:

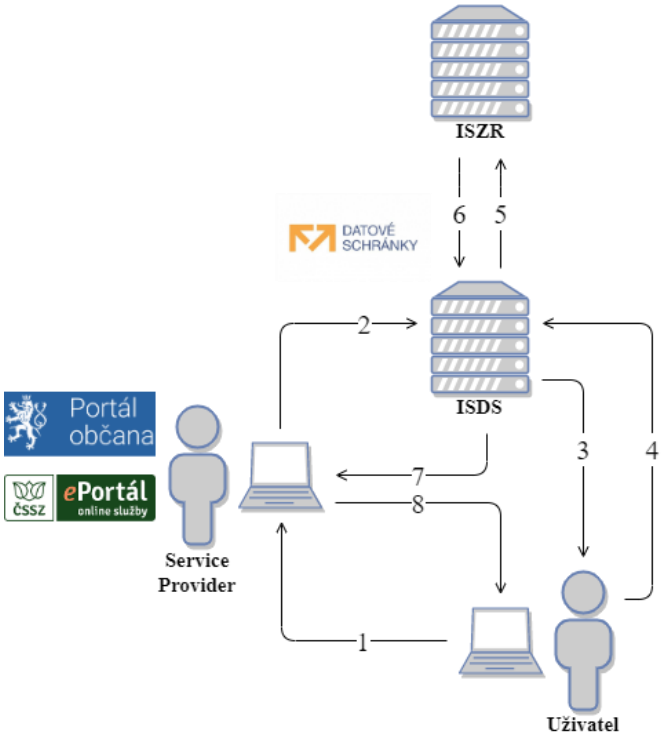
- SeP se nestará o:
  - metody přihlášení,
  - aktuálnost dat o uživateli,
  - spolehlivost IdP,
- uživatel vidí, o jaká data SeP žádá a dává souhlas s jejich použitím.

Nevýhodou zvoleného modelu může být spoléhání se na jeden bod v průběhu komunikace. V případě jeho kompromitace mohou uniknout osobní údaje a celá integrita transakce je narušena. NIA totiž zaznamenává veškerou provedenou komunikaci.

ISDS proti tomu využívá **přímého modelu**. Přímý model spočívá v tom, že ISDS ví, jakým způsobem se uživatel přihlašuje a je na samotném ISDS jaké metody přihlášení (kromě přihlášení pomocí eOP, které je stanoveno v zákoně č. 300/2008 Sb., *o elektronických úkonech a autorizované konverzi dokumentů* [24]) nabídne. Není využit prostředník v průběhu komunikace.

Nevýhoda tohoto modelu je přímá závislost na systému ISDS. Pokud by došlo k výpadku ISDS nebude možné se přihlásit k jakékoliv službě. Když dojde k výpadku u nepřímého modelu, např. poskytovatele karet Starcos, stále bude možné se přihlásit pomocí eOP.

Obrázek 8 - Přímý model ilustruje postup ověřování uživatele u přímého modelu.



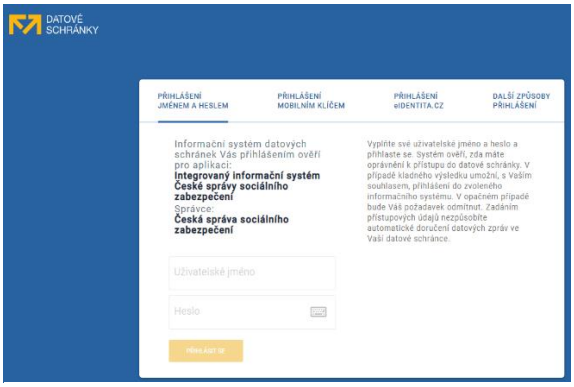
Obrázek 8 - Přímý model

Krok číslo	Popis
1	Uživatel vyžaduje službu od SeP.
2	Uživatel je přesměrován na ISDS.
3	ISDS nabídne uživateli možnosti přihlášení.
4	Uživatel se přihlásí jím zvolenou metodou.
5	V případě úspěšného ověření uživatele, je registrům odeslána žádost o požadované atributy.
6	ISZR vydává požadované atributy.
7	ISDS posílá SeP požadované atributy.
8	Uživatel je identifikován a autentizován a SeP uživateli poskytuje žádanou službu.

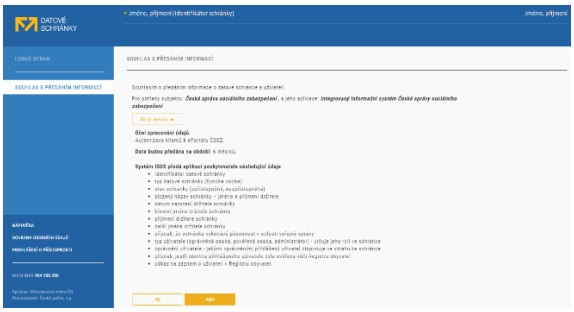
Tabulka 3 – Proces přímého modelu

Pro znázornění je proces ilustrován na přihlášení k ePortálu ČSSZ.

Obrázek 9 – Přímý model – Portál se službou



Obrázek 10 – Přímý model – Přihlášení k ISDS



Obrázek 11 – Přímý model – Souhlas s údaji

Obrázek 12 – Přímý model – Zpřístupněná služba

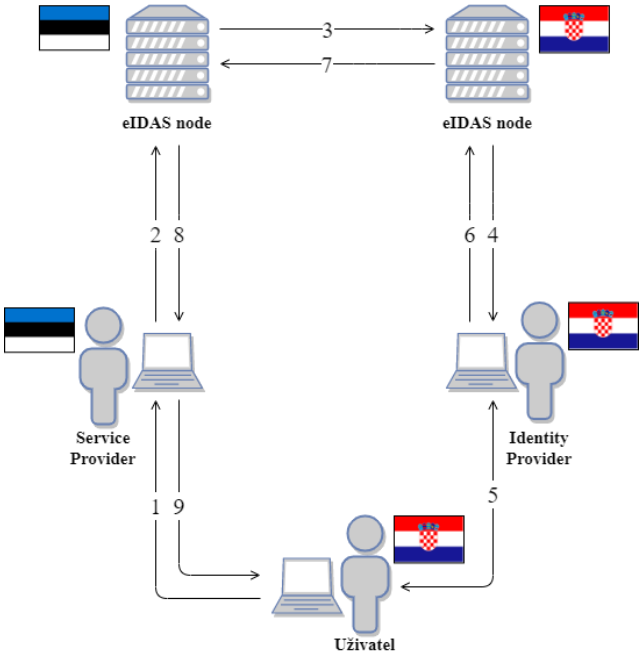
Obrázek číslo	Popis
8	Uživatel vyžaduje službu od SeP.
9	Uživatel je přesměrován na portál DS, kde si může vybrat metodu přihlášení.
10	Uživatel je po úspěšném ověření vyzván k odsouhlasení výdeje údajů.
11	Pokud vše proběhlo správně, zpřístupní se požadované služby (např. Přehled dob důchodového pojištění nebo Informativní výpočet starobního důchodu).

Tabulka 4 - Znázornění přímého modelu

3 Elektronická identifikace v evropském kontextu

Evropský parlament a Evropská rada vytvořily nařízení eIDAS mimo jiné proto, aby byl zajištěn **jednotný digitální trh**. S tím se pojí i **usnadnění přístupu** k zahraničním online službám nebo přeshraniční uznávání elektronické identifikace [1]. Pro obyvatele členského státu EU je možné využívat služby veřejnoprávního SeP i mimo stát ve kterém byl vydán prostředek pro elektronickou identifikaci. Tyto a další služby má právně zajistit Evropská komise, která odpovídá za předkládání návrhů nové evropské legislativy [25].

Celý koncept zobrazuje následující obrázek.



Obrázek 13 – Komunikace mezi eIDAS uzly

Krok číslo	Popis
1	Chorvatský uživatel, žijící v Estonsku, žádá službu po estonském SeP.
2	SeP nabízí možnost ověřit se i cizím státním příslušnostem. Po kliknutí na mezinárodní přihlášení, začne komunikovat s národním eIDAS uzlem.
3	Uživatel si vybere, jaký mezinárodní uzel má být kontaktován.

4	Uživatel je přesměrován na výběr IdP, který mu vydal prostředek pro elektronickou identifikaci.
5	Mezi uživatelem a IdP proběhne ověření.
6	IdP posílá zprávu o úspěšné autentizací národnímu eIDAS uzlu.
7	Uzly si předají zprávu.
8	SeP je doručena zpráva s úspěšným ověřením uživatele.
9	Uživateli je zpřístupněna požadovaná služba.

Tabulka 5 - Komunikace mezi eIDAS uzly

3.1 Podmínky nařízení eIDAS

Nařízení eIDAS stanovuje podmínky, které je nutné splnit dříve, než bude vzájemné uznávání nástrojů elektronické identifikace mezi členskými státy možné.

- První z nich je, že daný **prostředek pro elektronickou identifikaci musí být vydán v rámci systému elektronické identifikace, který je uveden na seznamu zveřejněném Komisí** podle článku 9 nařízení eIDAS (Oznámení) [1]. Aktuální seznam vydaný 8.4.2020 čítá 15 států, které mají zveřejněné systémy elektronické identifikace [26]. Dne 13.9.2019 se na seznamu uznaných systémů objevil Vnitrostátní systém identifikace České republiky. Postup oznámení systému elektronické identifikace a časové rámce uznávání oznámených prostředků je uveden v následující kapitole 3.2 Interoperabilita.
- Další podmínky se týkají LoA. **Obě strany**, SeP i IdP**, musí využívat úroveň značnou nebo vysokou**. Uznávání prostředků pro elektronickou identifikaci s LoA nízká je čistě záležitost SeP. Uznávat tento prostředek není povinností, ale prostředek musí být vydán v rámci kvalifikovaného systému, který je uveden na seznamu zveřejněném Komisí. Tyto podmínky platí pouze pro subjekty veřejného sektoru.

3.2 Interoperabilita

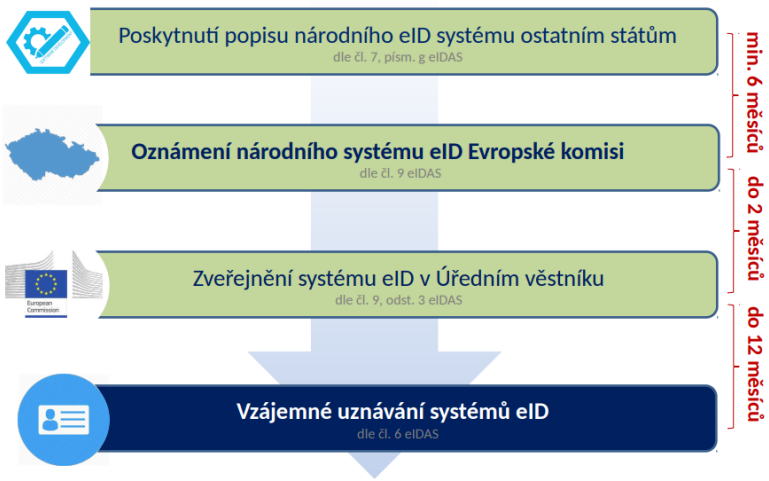
Pro splněný požadavků na interoperabilitu je zaveden **rámec interoperability**. Rámec má zajistit integritu a správnost dat. Dle tohoto rámce je nutné dodržet normu ISO/IEC 27001 při

stavbě uzlu eIDAS. Uzel by měl od počátku svého vzniku dbát na bezpečnost a ochranu soukromí. Jednotlivé uzly **nesmí klást nepřiměřené technické nároky** pro dosažení interoperability [20]. Při vypracování rámce interoperability byl maximálně zohledněn **projekt STORK**.

STORK se snažil o propojení a standardizaci identit mezi státy EU. Bohužel měl nedostatečný právní základ a nebylo povinné ho zavést [27]. Ovšem uzel, který byl vytvořen v projektu STORK 2, s názvem **CZ.PEPS**, jehož provoz má na starosti **CZ.NIC**, je základní stavební kámen pro **uzel eIDAS** [28].

Zkoumání principu interoperability a LoA se provádí pomocí prováděcích aktů, které jsou přijímány přezkumným postupem. Do přezkumného postupu je zapojen výbor, složený ze zástupců členských zemí, který vydává kladné nebo záporné stanovisko na prováděcí akt. Na základě tohoto stanoviska se pak řídí Komise. Přezkumný postup se používá zejména při prováděcích aktech s obecnou působností nebo při prováděcích aktech s programem se značným dopadem [29].

Konkrétním dnem, kdy vstoupil v platnost článek 6 (Vzájemné uznávání) nařízení eIDAS, je 29.9.2018. Od té doby je pro členské země povinné uznávat prostředky pro elektronickou identifikaci, které jsou ohlášené Komisí. Ohlašující země musí poskytnout popis svého systému a ostatní členské státy mají 6 měsíců na případné připomínky. Pokud nejsou shledány žádné chyby, je systém oznámen Komisi a ta ho musí do 2 měsíců uvést v Úředním věštníku. Členské země mají až 1 rok na to, aby zajistily, že ohlášené prostředky pro elektronickou identifikaci budou použitelné v jejich zemi. Časový postup je znázorněn na Obrázek 14 - Časové schéma [28].



Obrázek 14 - Časové schéma [28]

Pro zanesení systému elektronické identifikace do seznamu zveřejňovaného Komisí je nutné do oznámení uvést požadované informace [1]:

- popis systému elektronické identifikace, včetně LoA,
- popis vydavatele prostředků pro elektronickou identifikaci v rámci systému,
- režim dohledu a informace o režimu odpovědnosti,
- orgán odpovědný za systém elektronické identifikace,
- informace o subjektu, který spravuje evidenci osobních identifikačních údajů,
- popis způsobu, jakým jsou plněny požadavky interoperability,
- popis online autentizace oznamujícího členského státu a zajištění dostupnosti online autentizace způsobem, aby spoléhající služba v jiném členském státě, mohla potvrdit osobní identifikační údaje,
- opatření k pozastavení platnosti nebo zrušení oznámeného systému elektronické identifikace nebo autentizace.

NIA již oficiálně podporuje několik zahraničních prostředků pro elektronickou identifikaci.

Obrázek 15 – Přihlášení k uzlům eIDAS

3.3 Problémy

V současné době jsou známy některé problémy s mezinárodním uznáváním prostředků pro elektronickou identifikaci.

- Mezinárodní prostředky mají být uznávány v online službách veřejného sektoru. Nikde však není psáno, že členský stát musí mít zavedenou elektronickou identifikaci pro eGovernment.
- Pokud ji zavedenou má, tak stále nemusí přijímat jiné prostředky, protože povinnost uznávat cizí prostředky se vztahuje na systémy, které jsou uvedeny v Úředním věštníku Evropské unie. Ohlášení systému je věc nepovinná. To vše navzdory tomu, že uzel eIDAS je povinný pro každý členský stát [30].
- Problém také je, že samotné přeshraniční přihlašování musí podporovat i vnitrozemní SeP, tedy že by měl umět zpracovat údaje o cizinci [31].

4 Vysoké školy

Jak již bylo psáno výše, pro vystupování jako SeP je potřebná DS OVM (nebo výjimka ze zákona). Vysoké školy jsou ze zákona č. 111/1998 Sb., *o vysokých školách*, brány jako právnické osoby [22]. DS PO však znemožňují vystupovat jako SeP, tedy vysokým školám je znemožněno poskytovat online služby v rámci NIA. Změna nastala s novelou zákona č. 300/2008 Sb., *o elektronických úkonech a autorizované konverzi dokumentů*. Novela přidělila vysokým školám práva a povinnosti jako OVM [24]. Konkrétní změna nastala v §1, kde je nahrazena definice OVM z původní:

„*Tento zákon upravuje elektronické úkony státních orgánů, orgánů územních samosprávných celků, státních fondů, zdravotních pojišťoven, Českého rozhlasu, České televize, samosprávných komor zřízených zákonem, notářů a soudních exekutorů*“

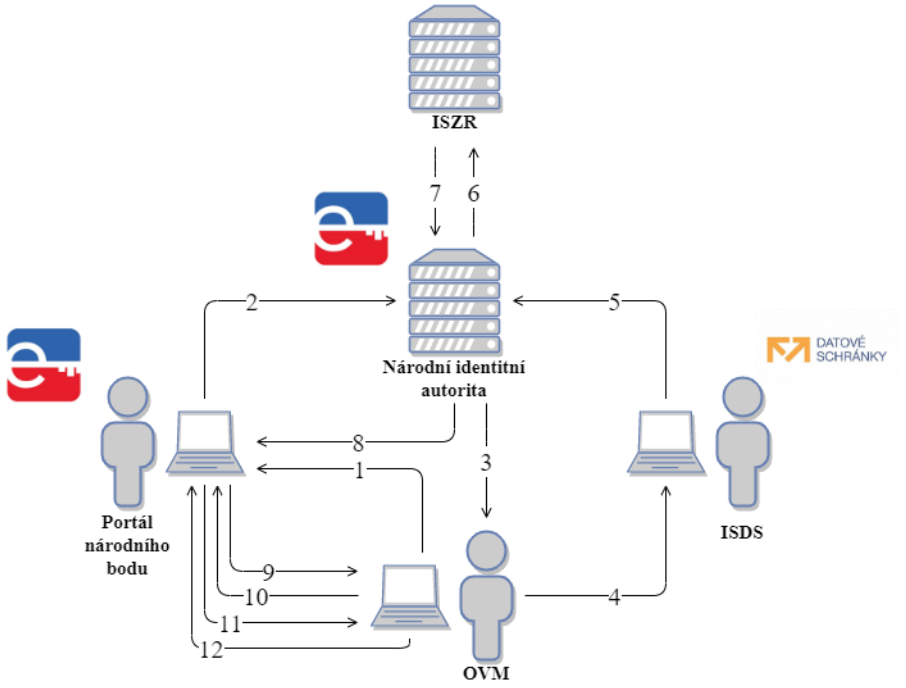
na novou: *„Tento zákon upravuje elektronické úkony státních orgánů, územních samosprávných celků a fyzických nebo právnických osob, pokud těmto fyzickým nebo právnickým osobám byla svěřena působnost v oblasti veřejné správy, (dále jen „orgán veřejné moci“)“*.

Podle definice zákona je vysokým školám svěřena působnost v oblasti veřejné správy, konkrétně školství.

Zákon č. 111/2009 Sb., *o základních registrech*, vymezuje pojem OVM stejným způsobem jako zákon č. 300/2008 Sb., *o elektronických úkonech a autorizované konverzi dokumentů* [24; 32]. Vysokým školám tak byly zřízeny nebo změněny DS PO na DS OVM-PO. Vzhledem k těmto dostupným informacím je možné vysoké školy zaregistrovat jako SeP.

Postup, jak zaregistrovat OVM jako SeP, je vidět na Obrázek 16 - Registrace .

Teoretická část touto kapitolou končí, protože splnila vytyčené cíle.



Obrázek 16 - Registrace OVM jako SeP

Krok číslo	Popis
1	OVM, resp. jeho vedoucí (u vysoké školy rektor nebo jím pověřená osoba [22]) žádá po portálu národního bodu registraci. Portál národního bodu v tuto chvíli vystupuje jako SeP, protože poskytuje službu registrace individuálních SeP.
2	Portál národního bodu kontaktuje NIA a požaduje ověření OVM.
3	NIA přesměruje OVM na ISDS.
4	ISDS plní roli IdP. OVM se přihlásí k DS, aby došlo k ověření.
5	V případě úspěšného ověření OVM odesílá ISDS autentizační token NIA. Token obsahuje identifikační údaje.
6	NIA požaduje atributy o OVM od ISZR.
7	ISZR vydává požadované atributy. Zároveň je provedena kontrola existence IČO.
8	NIA posílá atributy (sesbírané od ISDS a ISZR) SeP.
9	SeP nabízí registraci OVM s předvyplněným formulářem.
10	OVM potvrdí údaje a registruje se.
11	SeP zpracuje registraci OVM. OVM je v tuto chvíli povolena konfigurace kvalifikovaného poskytovatele.
12	OVM provádí konfiguraci svých služeb. Potřebné údaje jsou uvedeny v kapitole 2.1.2 Kvalifikovaný poskytovatel online služeb.

Tabulka 6 - Registrace OVM jako SeP

5 Návrh systému

V teoretické části již bylo probráno, jak funguje NIA i to, že vysoké školy mohou využívat jejích služeb. Tím je umožněno přistoupit k praktické části. Pro zvýšení přehlednosti jsou zopakovány cíle praktické části:

- popis existujících softwarových modulů pro implementaci služeb Národní identitní autority,
- vytvoření studie proveditelnosti implementace služeb Národní identitní autority do informačního systému vysoké školy.

Praktická část je dále dělena na **návrhovou** a **implementační část**.

Návrhová část se zabývá zejména:

- **návrhem architektury**,
- **popisem standardu SAML,**
- **výběrem autentizačního protokolu** a jeho popisem,
- **výběrem softwarových nástrojů** a jejich popisem.

Pro sestavení systému je důležité definovat, kdo je primárním uživatelem vysokoškolského systému v této práci. **Hlavním uživatelem je uchazeč o studium**, který musí ověřit svoji identitu při podání přihlášky, **a dále student**, vykonávající denní rutinu v informačním systému. **Pro uchazeče o studium** je potřeba zajistit **přihlášení skrze NIA**, kdežto pro již studujícího může být tento proces nadbytečný. Vyplývá z toho potřeba **alespoň dvou způsobů přihlášení**.

Pro zobecnění, **informační systém** představuje **webový portál OVM** s nabídkou služeb a **uživatelem je občan**, který žádá o přístup ke službám.

Informační systém vysoké školy není nutné replikovat do detailu, důležité je zajistit a ověřit jeho základní funkčnost. Tento informační systém je webová aplikace, pro kterou je třeba zajistit:

- **open source software**,

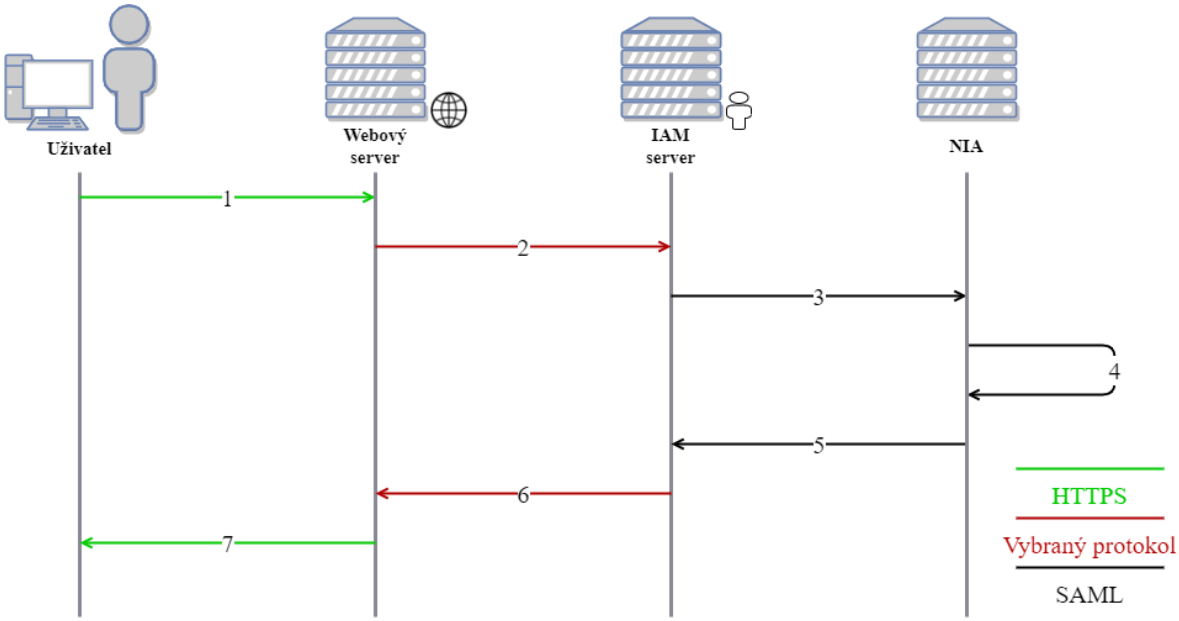
- hardware (v podobě virtuální privátního serveru (VPS)).

Výběr serveru je rozebrán v kapitole 9 Aplikační server. Výběr VPS není významnou součástí práce, proto je v kapitole pouze krátce zmíněn.

Pro správu identit a oprávnění se nabízí použít open source software, jehož hlavní funkce je **Identity & Access Management** (IAM, systém pro správu identit a přístupu). IAM software by měl správci nabízet přehledné a moderní webové prostředí s množstvím funkcí. Ideální vybíraný IAM software nabízí, kromě správy identit, další možnosti jako:

- volbu metod přihlášení přistupujícího uživatele,
- správu hesel,
- podporu standardu SAML (kapitola 6 SAML 2.0) a dalšího vybraného protokolu (kapitola 7 Autentizační protokol).

Navrhovaná architektura je obecně zobrazena na Obrázek 17 - Navrhovaná architektura.



Obrázek 17 - Navrhovaná architektura

Krok číslo	Popis
1	Uživatel si chce zobrazit stránky s informačním systémem.

2	Pokud požaduje přístup ke službám je povinen se přihlásit. V tuto chvíli je přesměrován na přihlašovací stránku, kterou poskytuje IAM server (server s IAM softwarem).
3	V případě, že se jedná o přihlášení, kdy je nutná spolehlivá identifikace osoby, je uživatel přesměrován na NIA. V případě, že se jedná o rutinní přihlášení, stačí vyplnit např. jméno a heslo. IAM server odesílá zprávu o úspěšném přihlášení webové aplikaci – krok č. 6.
4	Uživatel se ověří vůči NIA. Celý postup je popsán v kapitole 6 SAML 2.0.
5	IAM server dostane odpověď. Data o uživateli jsou uložena a webovému serveru jsou pomocí vybraného protokolu předána.
6	Webový server přijme a zpracuje odpověď.
7	Uživateli je zobrazena stránka o úspěšném přihlášení a zobrazeny dostupné služby.

Tabulka 7 - Navrhovaná architektura

5.1 Komponenty

Navrhovaný systém se skládá z následujících komponent:

- protokolu SAML,
- autentizačního protokolu,
- IAM software,
- aplikačního serveru.

V dalších kapitolách jsou všechny části systému rozebrány.

6 SAML 2.0

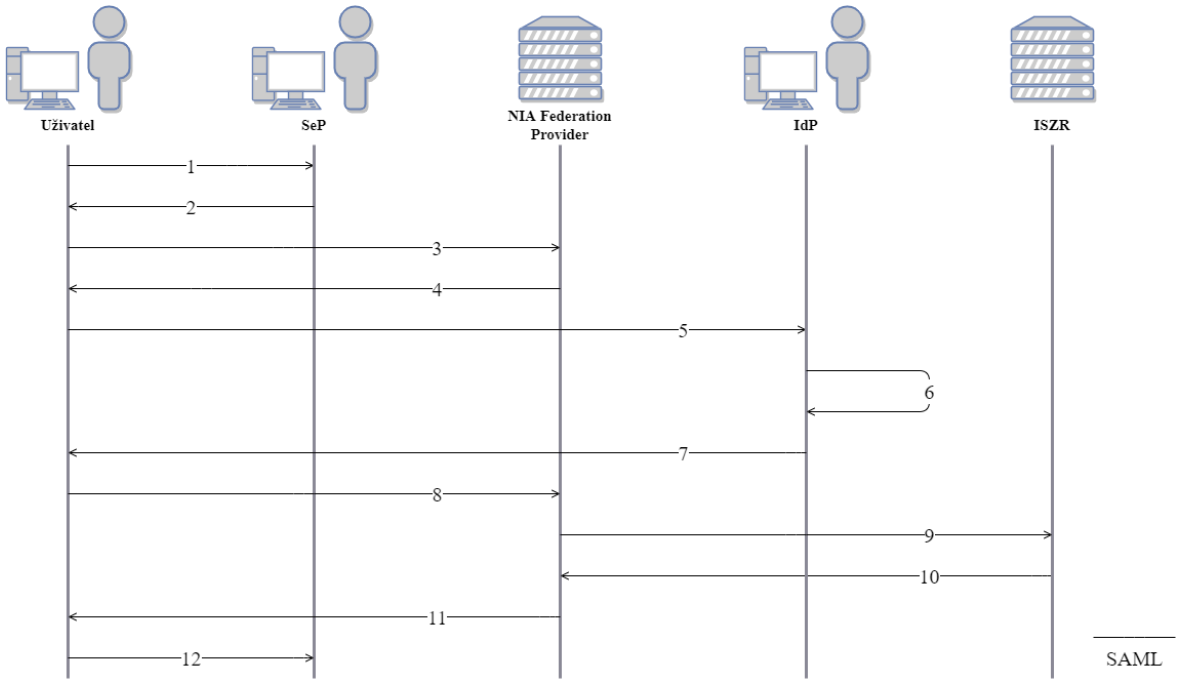
NIA pro komunikaci se SeP a IdP používá protokol SAML (Security Assertion Markup Language). V návrhové části je první popisovanou komponentou, protože je pevně dáno jeho použití.

SAML verze 2.0 je standard pro výměnu autentizačních a autorizačních dat, který byl vydán v roce 2005 a je založen na značkovacím jazyce XML. Atributy jsou přenášeny v SAML Response, respektive SAML Assertion, mezi aplikacemi, které mají vytvořený vztah důvěry [3; 17]. SAML je především používán na webových stránkách a využití najde hlavně mezi organizacemi. Standard pracuje s pojmy SeP a IdP, podobně jako nařízení eIDAS.

Mezi hlavní výhody protokolu SAML patří:

- Single Sign-On (SSO) [33],
- široká webová podpora,
- vyspělost získaná dlouholetým používáním.

Na následujícím obrázku je vidět, jak probíhá přihlašování u NIA s protokolem SAML.



Obrázek 18 - Přihlašovací proces

Krok číslo	Popis
1	Uživatel požaduje po SeP službu.
2	SeP posílá uživateli SAML Request s požadovanými atributy.
3	Uživatel přeposílá SAML Request NIA Federation Providerovi (součást systému NIA). Ten zobrazí dostupná IdP podle požadované LoA.
4	NIA Federation Provider přesměruje uživatele na zvoleného IdP.
5	Uživatel se ověří vůči IdP.
6	IdP zkontroluje platné přihlašovací údaje a vytvoří SAML Response.
7	Uživateli je předána SAML Response.
8	Uživatel předá platnou SAML Response NIA Federation Providerovi.
9	NIA Federation Provider si vyžádá atributy od ISZR.
10	ISZR posílá zpět získané atributy o uživateli.
11	Uživateli je zaslána SAML Response, ve které je zakomponována SAML Assertion s atributy.
12	Uživatel předává SAML Response SeP. Ten jí rozšifruje a použije získané atributy.

Tabulka 8 - Přihlašovací proces

V následujících podkapitolách, jsou rozebrány komponenty protokolu SAML – Request, Response, Assertion a metadata. Využívané elementy a jejich atributy jsou v souladu s nařízením eIDAS, ale je možné žádat i atributy specifické pro NIA. Kompletní ukázky je možné nalézt v Příručce k využití služeb NIA [17].

6.1 SAML Request

SeP musí vygenerovat požadavek o identifikaci a v něm specifikovat:

- LoA,
- atributy o uživateli, která požaduje v návratovém tokenu [17].

Rozebíraná ukázka je dostupná zde.

Elementy/atributy	Popis
<saml2p:AuthnRequest> ID	Specifikace žádosti ID zprávy

Destination AssertionConsumerServiceURL	Adresa, na kterou je žádost odeslána Adresa pro příjem tokenu
<saml2Issuer>	Unikátní adresa
<saml2p:RequestedAuthnContext> Comparison="minimum"	Element obalující specifikaci LoA Požadavek na nejnižší LoA
<saml2:AuthnContextClassRef>	Samotná specifikace LoA
<eidas:SPType>	Identifikace veřejného nebo soukromoprávního SeP
<eidas:RequestedAttributes>	Seznam požadovaných atributů
<eidas:RequestedAttribute> Name isRequired	Jednotlivé požadované atributy Název atributu podle dokumentace Specifikace, zda SeP konkrétní atribut potřebuje

Tabulka 9 - SAML Request

V následující tabulce jsou uvedené atributy, o která může SeP v SAML Request požádat. **Převod těchto atributů do modernějšího standardu je důležitou součástí práce**. Tučně zvýrazněné atributy odpovídají nařízení eIDAS, ostatní jsou záležitostí NIA.

Uživatelské atributy	Popis
CurrentFamilyName	Příjmení
CurrentGivenName	Jméno
DateOfBirth	Datum narození
PlaceOfBirth	Místo narození
CountryCodeOfBirth	Kód země narození
CurrentAddress	Adresa pobytu zakódovaná pomocí Base64
Email	Emailová adresa
IsAgeOver	Výpočet je starší než
Age	Věk
PhoneNumber	Telefonní číslo
TRadresaID	Adresa pobytu zakódovaná pomocí Base64
LoA	Stupeň jistoty
PersonIdentifier	Identifikátor fyzické osoby
IdType	Druh elektronicky čitelného dokladu

IdNumber	Číslo elektronicky čitelného dokladu

Tabulka 10 – Atributy

CurrentAddress a TRadresaID jsou složené atributy a částečně se překrývají v obsahu (PSČ a číslo orientační). Jsou mezi nimi však rozdíly.

- CurrentAddress odpovídá standardu eIDAS, kdežto TRadresaID je záležitostí NIA.
- Další rozdíl je, že v atributu CurrentAddress jsou uváděny názvy jednotlivých atributů, ale v atributu TRadresaID jde pouze o kódy podle Registru územní identifikace, adres a nemovitostí (RÚIAN).

CurrentAddress obsahuje atributy:

- název ulice (Thoroughfare),
- název pošty (PostName),
- PSČ (PostCode),
- název obce (CvaddressArea),
  - případně část obce,
- číslo orientační (LocatorDesignator).

TRadresaID obsahuje atributy:

- kód pro ulici,
- kód pro okres,
- kód pro obec,
- kód pro část obce,
- PSČ,
- kód pro stavební objekt,
- kód pro adresní místo,
- číslo domovní a orientační.

Zákon č. 111/1998 Sb., *o vysokých školách*, konkrétně § 50 odst. 1 specifikuje minimum údajů, které musí uchazeč o studium vyplnit [22]. Jsou to:

- **jméno**, odpovídá atributu **CurrentGivenName,**
- **příjmení**, odpovídá atributu **CurrentFamilyName,**
- rodné číslo, neodpovídá žádnému atributu**,**
- **adresa**, odpovídá atributu **CurrentAddress,**

Získat tyto atributy je stěžejní.

6.2 SAML Response

Rozebíraná je odpověď od IdP, která obsahuje SAML Assertion. Popisovaná ukázka je dostupná zde.

Elementy/atributy	Popis
<saml2p:Response> Destination InResponseTo	Specifikace žádosti Adresa, kam je token doručen ID původní žádosti
<saml2Issuer>	Identifikátor IdP
<ds:Signature>	Podpis veřejným certifikátem IdP
<saml2p:Status>	Informace o stavu žádosti
<saml2:EncryptedAssertion>	Zašifrované atributy o uživateli

Tabulka 11 - SAML Response

6.3 SAML Assertion

Samotné uživatelské atributy se skrývají v SAML Assertion. SAML Assertion je zakódováno pomocí Base64 a pro dešifrování jednotlivých atributů, je nutné použít soukromý klíč, který vlastní SeP. Rozebíraná ukázka je dostupná zde.

Elementy/atributy	Popis
<saml:Assertion>	Specifikace SAML Assertion
<saml:NameID>	Identifikátor uživatele
<SubjectConfirmation Method="… bearer">	Metoda, která se používá pro potvrzení správnosti dat. Zde typ bearer (nosič).
<SubjectConfirmationData InResponseTo NotOnOrAfter Recipient>	ID odpovědi Časový údaj, po kterém není možné brát data jako potvrzená. Příjemce SAML Assertion
<Conditions NotBefore="…" NotOnOrAfter="…">	Podmínky, za jakých je možné SAML Assertion použít. Časové údaje, během kterých je možné brát SAML Assertion jako platnou.
<Attribute>	Definice, o jaký atribut se jedná.
<AttributeValue>	Hodnoty samotných požadovaných atributů.

Tabulka 12 - SAML Assertion

6.4 Metadata

Metadata jsou data, která jsou veřejně vystavená proto, aby bylo možné automaticky navázat potvrzenou komunikaci. Metadata zveřejňují veřejné klíče, endpointy (důležité pro SSO), informace o organizaci, kontrolní digesty (hashe) a také jaký algoritmus je použit pro hashovaní. NIA má testovací metadata zveřejněná zde.

Elementy/atributy	Popis
<EntityDescriptor>	Jednoznačný identifikátor SAML entity
<Signature>	Informace o klíči a určení způsobu hashování
<SignatureMethod>	Určení algoritmu hashování pro podpis
<DigestMethod>	Specifikace algoritmu hashování pro digest.
<DigestValue>	Hash podepisovaného zdroje.
<SignedValue>	Podepsaná hodnota elementu <SignedInfo>.
<X509Certificate>	Veřejný klíč odesílatele podepsaný certifikační autoritou.
<IDPSSODescriptor>	Popis služeb poskytujících SSO.
<SingleLogoutService/SingleSignOnService> Binding Location	Specifikace jednotlivých endpointů SSO. Zvolená metoda přesměrování dat. Adresa endpointu.

Tabulka 13 - SAML Metadata

6.4.1 Klíče

Vystavení veřejného klíče je důležité pro **princip elektronického podpisu**. Ten má zajistit integritu a důvěrnost dat. Podpis se skládá ze dvou částí – **veřejného** a **soukromého klíče**. Veřejný klíč je vystaven, soukromý klíč pak drží pouze vydávající strana. Odesílatel nejdříve zašifruje spočítaný hash svým privátním klíčem (digitální podpis) a poté zašifruje zprávu veřejným klíčem příjemce. Příjemce rozšifruje zprávu svým privátním klíčem a veřejným klíčem odesílatele digitální podpis. Na rozšifrovanou zprávu použije stejný algoritmus hashování jako odesílatel a pokud se výsledná hodnota shoduje s hodnotou hashe v digitálním

podpisu, je jistota, že zpráva nebyla v průběhu komunikace kompromitována. Hashe jsou používány pro ušetření výkonu [34].

7 Autentizační protokol

Autentizační protokol je využíván pro komunikaci mezi webovým a IAM serverem. Podstatnou podmínkou, kterou musí protokol splňovat, **je modernější podoba dat**. Původní specifikace jazyka XML, kterým se přenáší uživatelské atributy z NIA, byla vydána roku 1998, nejnovější 5. edice vyšla roku 2008 [35]. XML je robustní, ale pro přenos uživatelských atributů se lépe hodí formát **JSON**. JSON je srozumitelnější, jednodušší a především modernější (specifikace vyšla roku 2017 [36]). V současné době jde také o **používanější standard** [37].

Základními předpoklady pro autentizační protokol jsou:

- **používání formátu JSON**,
- **použitelnost v mobilních i webových aplikacích**.

Dostupné jsou tyto protokoly:

- Kerberos,
- OpenID Connect,
- WS-(Federation, Security).

Požadavky	**Autentizační protokol**		
	Kerberos	**OpenID Connect**	WS-(Federation, Security)
JSON	Ne	**Ano**	Ne
Datum vydání	2005 (verze 5)	**2014 (verze 1.0)**	Federation 2009 (verze 1.2), Security 2004 (verze 1.1)
Mobilní i webové aplikace	Ne	**Ano**	Ne

Tabulka 14 – Výběr autentizačního protokolu

Protokol pro autentizaci s požadovanými vlastnostmi je **OpenID Connect**.

7.1 OpenID Connect

Protokol OpenID Connect (OIDC) je spravován organizací OpenID Foundation. Specifikace protokolu OIDC 1.0 vyšla v roce 2014 a je tedy novější než SAML 2.0 (2005) [38]. Umožňuje ověření identity uživatele pomocí přihlášení na vzdáleném serveru a podporuje SSO. Přenášená data jsou ve formátu **JSON**. Protokol OIDC i formát JSON jsou vytvořeny tak, aby podporovaly mobilní i webové aplikace. V jedné z rozšiřujících specifikací protokolu OIDC je zmíněno, že je s nařízením eIDAS kompatibilní [39]. Některé členské země EU již protokol OIDC v kombinaci s nařízením eIDAS využívají. Jde např. o:

- Finsko,
- Norsko,
- Francii.

Mezi organizace využívající OIDC se řádí např.:

- Apple,
- IBM,
- Red Hat,
- mojeID.

Pro správné pochopení OIDC jsou v tabulce vysvětleny základní pojmy [38; 40].

Pojem	Popis
Relying Party (RP)	Aplikace, která požaduje autentizaci. Podobný pojem jako SeP u NIA a protokolu SAML. Rozdíl mezi RP u OIDC a SeP u SAML je ten, že SeP je vždy webová stránka, ale RP může být webová i mobilní aplikace.
OpenID Provider (OP)	Poskytovatel identit. Srovnatelný s IdP u NIA a protokolu SAML.
Claims	Atributy, které jsou vyžadovány/posílány. Základní uživatelské atributy jsou v tabulce níže. Možné je vytvořit i vlastní.
ID Token	Token, který je zasílán jako **JSON Web Token** (JWT). Obrazně je možné ID Token připodobnit občanskému průkazu. V ID Tokenu mohou být zasílány claims. ID Token může vypadat např. takto:

	• "iss" – URL adresa serveru, který ID Token vydal, • "sub" – identifikátor uživatele, • "aud" – client_id u RP, • "nonce" – kód, který má chránit před zneužitím tokenu třetí osobou, • "exp" – časový údaj po kterém token není brán jako validní, • "iat" – časový údaj, kdy byl token vytvořen, • "auth_time" – časový údaj, kdy byl uživatel autentizován, • "acr" – hodnota, jak silná byla použita autentizace.
Access Token	Přístupový token, díky kterému je možné přistoupit k UserInfo Endpointu.
Scope	Skupiny claims, o které je možné žádat. Předdefinované Scopes jsou: • profile, • email, • address, • phone. Je možné vytvořit i vlastní Scope.
Endpoint	Bod, adresa či umístění na kterých je možné zažádat o tokeny. Mezi definované patří: • Authorization Endpoint – adresa pro přihlášení a autorizaci. • Token Endpoint – vydává ID a Access Token. • UserInfo Endpoint – poskytuje claims o uživateli.
Flows	Komunikační toky, které protokol využívá. Definované jsou tyto: • Authorization Code Flow – nejbezpečnější z uvedených Flows. Z Authorization Endpointu je vrácen Authorization Code, který je poté využit pro získání ID Tokenu a Access Tokenu. Tokeny nejsou zobrazeny uživateli. Vyžadováno HTTPS na obou stranách komunikace. • Implicit Flow – méně bezpečný než Authorization Code Flow. Není využíván Token Endpoint. Access Token je zobrazený v URL adrese. • Hybrid Flow – kombinace dvou předchozích Flows.

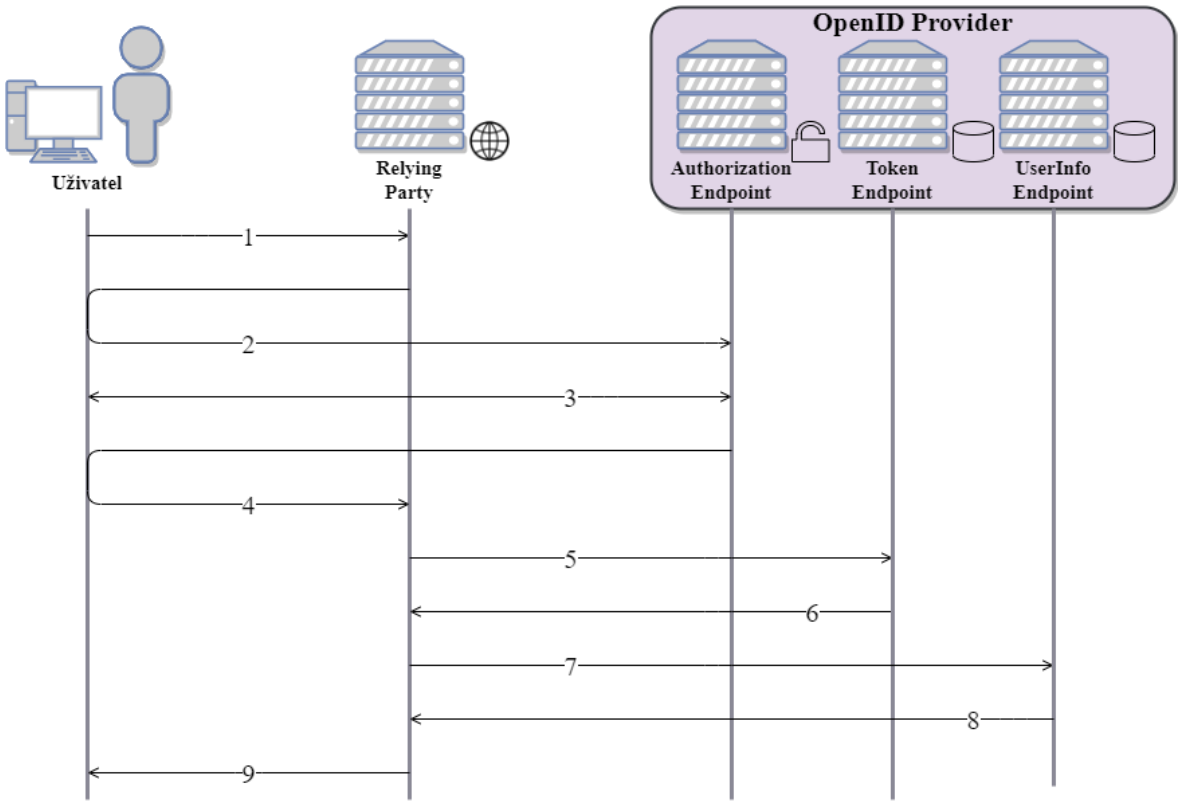
Tabulka 15 - OIDC pojmy

V následující tabulce jsou uvedené claims, které může RP vyžádat. **Právě na tyto claims je třeba zajistit převod vyžádaných atributů od NIA**.

Claims	Popis
sub	Identifikátor uživatele na straně OP
name	Celé jméno včetně titulů
given_name	**Křestní jméno**
family_name	**Příjmení**
middle_name	Druhé jméno
nickname	Přezdívka
preferred_username	Zkrácené jméno používané na straně RP
profile	URL adresa na stránku o uživateli
picture	URL adresa na obrázek uživatele
website	URL adresa webové stránky na kterou uživatel publikuje články
email	Emailová adresa
email_verified	Informace, zda byla emailová adresa ověřena
gender	Pohlaví
birthdate	Datum narození ve formátu YYYY-MM-DD
zoneinfo	Časová zóna
locale	Region a jazyk
phone_number	Telefonní číslo
phone_number_verified	Informace, zda bylo telefonní číslo ověřeno
address	JSON struktura, která obsahuje: • formatted – poštovní adresa, která se zobrazí, • street_address – kompletní poštovní adresa, • locality – město, • region – kraj, • postal_code – PSČ, • country – země.
updated_at	Čas, kdy byly naposledy aktualizovány informace o uživateli

Tabulka 16 - OIDC claims

Na následujícím obrázku je ilustrován **Authorization Code Flow**, který je vybrán pro použití v komunikaci mezi vytvořenou webovou aplikací a IAM serverem. Rozhodujícím faktorem výběru je jeho vyšší bezpečnost.



Obrázek 19 - OIDC Authorization Code Flow

Krok číslo	Popis
1	Uživatel navštíví webovou nebo mobilní aplikaci.
2	RP metodou HTTP POST nebo GET zašle žádost na Authorization Endpoint. V žádosti je nutné specifikovat: • host (location) – adresa, kam je žádost odeslána, • scope=openid – identifikace, že se jedná o protokol OIDC, možné specifikovat další požadované Scopes, • response_type=code – identifikace, že se jedná o Authorization Code Flow, • client_id – identifikace uživatele na straně autorizačního serveru, • redirect_uri – adresa, kam je zaslána odpověď, • state – identifikátor, který se používá pro zabránění útoku typu Cross-Site Request Forgery.

3	Proběhne přihlášení uživatele. Zároveň je uživatel vyzván k odsouhlasení přístupu k jeho identitě.
4	Authorization Endpoint posílá zpět odpověď s Authorization Code. V odpovědi jsou tyto údaje: • location – adresa, kam je odpověď zaslána (stejná jako redirect_uri v žádosti), • code – kód, kterým je možné zažádat o ID a Access Token, • state – stejný identifikátor jako v žádosti.
5	RP zažádá o ID a Access Token. RP posílá žádost přímo na Token Endpoint. Jedná o back-channel request, díky tomu je zajištěno, že tokeny nejsou zobrazeny prohlížeči, ale jen serveru. V žádosti jsou tyto parametry: • authorization – Base64 zakódované client_id a secret (kód, který zná jen RP a OP), • grant_type=authorization_code – informace o používané flow, • code – kód z předchozí odpovědi, • redirect_uri – adresa, stejná jako v první žádosti.
6	OP ověří správnosti client_id a secret, dále ověří správnost Authorization Code (a také to, zda už nebyl použit) a stejnou hodnotu redirect_uri jako v první žádosti. Pokud je vše správně Token Endpoint vydává odpověď s ID a Access tokeny. V odpovědi je uvedeno: • "access_token"– hodnota tokenu, • "id_token" – hodnota tokenu, • "token_type":"Bearer" – držitel tokenů je oprávněn se jimi prokazovat, • "expires_in" – časový údaj po který jsou tokeny platné. RP musí ověřit hodnoty ID Tokenu. Pokud vše sedí, je uživatelská identita ověřena. Uživatel je v tuto chvíli přihlášen.
7	Pokud RP žádá více claims o uživateli zašle Access Token na UserInfo Endpoint.
8	UserInfo Endpoint vybere požadované claims a zasílá je RP.
9	Claims jsou zobrazeny uživateli.

Tabulka 17 - OIDC Authorization Code Flow

7.1.1 OAuth 2.0

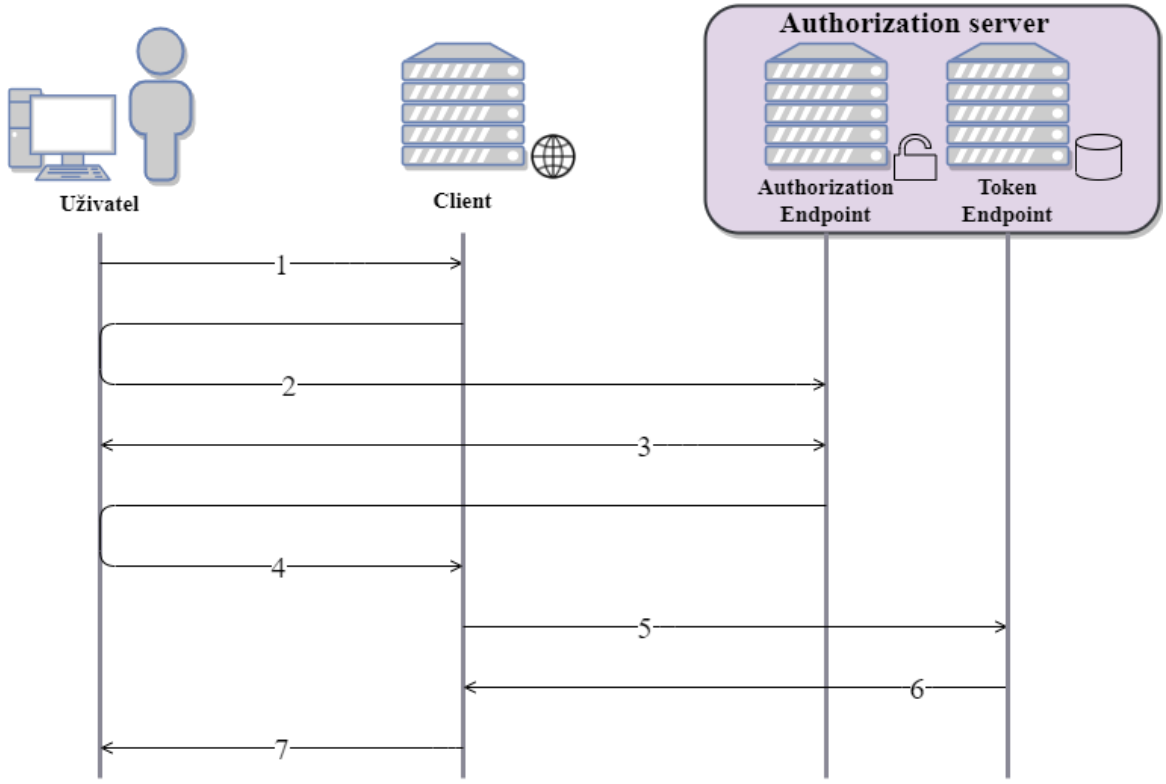
Pro doplnění je uveden protokol OAuth 2.0 na kterém OIDC staví. OAuth 2.0 je autorizační protokol definovaný v roce 2012. OIDC je v samotné specifikaci definován jako identitní

vrstva na protokolu OAuth 2.0 [38]. OAuth 2.0 široce používají služby, které žádají pouze omezená přístupová práva. Důležité je pochopit rozdíl mezi oběma protokoly. **OAuth 2.0 slouží pro autorizaci**, ale **OIDC pro autentizaci**. Zásadním rozlišením mezi protokoly je **ID Token** (viz Tabulka 15 - OIDC pojmy), který říká:

- kdo je uživatel,
- kdo je příjemce,
- kdo je vydavatel,
- jak dlouho je identita validní,
- jakou silou je uživatel ověřen.

Vývojáři aplikací často nevnímají rozdíl a používají OAuth 2.0 všude, ale to není správný přístup.

Pro pochopení je přítomný obrázek komunikace protokolu OAuth 2.0, konkrétně Authorization Code Grant (Flow).



Obrázek 20 - OAuth 2.0

Krok číslo	Popis
1	Uživatel přistupuje ke Clientu (aplikaci). Client pro správnou funkčnost vyžaduje práva od uživatele.
2	Client vyzve uživatele k autorizaci. Uživatel je poté přesměrován na Authorization Endpoint.
3	Uživatel se přihlásí k poskytovateli autorizace.
4	Přes uživatele je poslán Clientu Authorization Code.
5	Client pošle žádost o Access Token. K vydání použije hodnotu Authorization Code. Jedná se o back-channel komunikaci, tedy, že uživatel nevidí komunikaci.
6	Authorization Server přihlásí Clienta a vydá mu Access Token.
7	Uživateli je zpřístupněna požadovaná funkčnost na straně Clienta.

Tabulka 18 - OAuth 2.0

8 Identity & Access Management software

Další komponentou v návrhu je server s IAM softwarem. Před samotným výběrem je vysvětlen pojem IAM.

IAM je politika organizace, která má za cíl spravovat uživatelské identity systému a jejich přístupová práva k aktivům [41]. Správné nastavení politiky IAM dovoluje organizacím splňovat stanovené mezinárodní/národní/firemní zákony týkající se identit a uživatelských dat (např. GDPR, PCI DSS a HIPAA). Řízení IAM zahrnuje [41]:

- správu hesel,
- správu identit,
- správu přístupů,
- správu logů a událostí.

IAM software přináší organizacím hlavní výhodu, že je možné uvedené správy řídit z centrálního bodu.

8.1 Porovnání IAM software

Kromě výše uvedených základních předpokladů jsou hlavními požadavky pro vybraný IAM software:

- open source,
- podpora protokolů OIDC i SAML,
- možnost přidat poskytovatele uživatelských identit,
- federovaná autentizace,
- aktivní vývoj,
- možnost silné autentizace WebAuthn [7],

- vícefaktorová autentizace,
- SSO.

Mezi vybíraná open source IAM řešení jsou zařazeny produkty:

- CAS (Central Authentication Service),
- Gluu,
- Keycloak,
- OpenIAM,
- Soffid,
- WSO2.

Do výběru nebyly zahrnuty řešení Apache Syncope, Shibboleth Identity Provider a MidPoint, protože nenabízejí požadovanou funkčnost IAM softwaru. V těchto případech se jedná spíše o Identity Management, v podobě middleware, který se teprve kombinací s dalším softwarem stává plnohodnotným IAM řešením.

V tabulce níže je vidět porovnání vybraných dostupných IAM produktů. Tabulka je zpracována na základě dostupných online materiálů o produktu.

Požadavky	**IAM software**					
	CAS	**Gluu**	**Keycloak**	OpenIAM	Soffid	**WSO2**
OIDC	Ano	**Ano**	**Ano**	Ano	Ano	**Ano**
SAML 2	Ano	**Ano**	**Ano**	Ano	Ano	**Ano**
Federace identit	Ne	**Ano**	**Ano**	Ano	Ano	**Ano**
Přidání poskytovatele identit	Ano	**Ano**	**Ano**	Ano	Ano	**Ano**
Aktivní vývoj	Ano	**Ano**	**Ano**	Ano	Ano	**Ano**
Podpora WebAuthn	Ne	**Ano**	**Ano**	Ne	Ne (LinOTP)	**Ano**

Vícefaktorová autentizace	Ano	**Ano**	**Ano**	Ano	Ne (LinOTP)	**Ano**
SSO	Ano	**Ano**	**Ano**	Ano	Ano	**Ano**

Tabulka 19 - Výběr IAM softwaru

Z tabulky vyplývá, že požadavky splňují produkty Gluu, Keycloak a WSO2.

Gluu je spíše souborem více komponent, které tvoří IAM řešení. Gluu tedy nabízí rozsáhlé možnosti, ale nevyužití celého ekosystému Gluu je plýtvání potenciálem. Pro tuto práci se více hodí jiné řešení.

WSO2 zvládá všechnu požadovanou funkcionalitu, ale chyba tohoto řešení je, že bezpečnostní updaty **nejsou** open source. Bezpečnostní záplaty se dostanou až do nové verze a nejde je stáhnout a aplikovat v současné verzi. Mínusem je i starší administrátorské rozhraní.

Jako nejvhodnější se jeví řešení **Keycloak**.

8.2 Keycloak

Keycloak je IAM software vyvíjený společností Red Hat, avšak původně byl vyvíjen firmou JBoss (koupena Red Hatem). Red Hat vystupuje samostatně, ale po akvizici v roce 2019, je majitelem IBM. Red Hat využívá Keycloak pro svůj vlastní software RH-SSO, ale zároveň aktivně přispívá k vývoji Keycloak. S akvizicí JBoss získal Red Hat také software WildFly. WildFly je aplikační server na kterém je Keycloak postaven. Součástí WildFly je i webový server Undertow. WildFly, Undertow i Keycloak jsou vyvíjeny v **programovacím jazyku Java**.

Ke klíčovým vlastnostem Keycloak patří:

- **Identity Brokering** – důležitá služba pro tuto práci. Identity Brokering zajišťuje propojení SeP s několika IdP [42]. Jedna uživatelská identita tak může být propojena s více IdP.
- Podpora SSO.
- Podpora protokolů SAML, OIDC a OAuth 2.0.

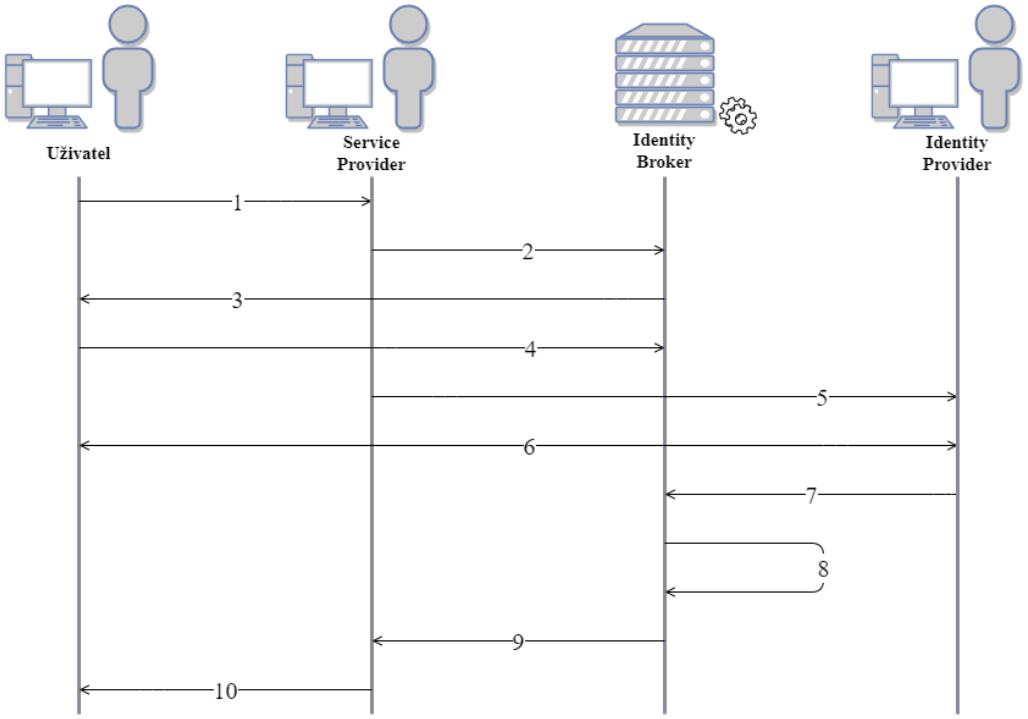
- Správa realms – realm je set uživatelů, aplikací, oprávnění a skupin, které přistupují ke zdroji.
- Logování událostí.
- Správa autentizace a heslové politiky.
- Správa oprávnění.
- Přehledné administrátorské prostředí.

8.3 Identity Brokering

Služba Identity Brokering propojuje více IdP s více SeP. Uživatel tak může svoji vytvořenou identitu u IdP (např. NIA, Facebook, Google, pracovní portály), využívat u SeP se kterými je navázán vztah důvěry. Jde o federovanou autentizaci. V Keycloak je možné použít 1 z protokolů:

- SAML,
- OIDC,
- OAuth 2.0.

Na Obrázek 21 - Identity Brokering je zobrazen graf, jak Identity Brokering funguje.



Obrázek 21 - Identity Brokering

Krok číslo	Popis
1	Uživatel chce využít služeb SeP, avšak musí se přihlásit.
2	SeP přesměruje uživatele na Keycloak. Zde je zobrazena jeho klíčová funkce **Identity Broker**.
3	Uživateli jsou zobrazeny IdP, kteří jsou nastaveni v **realm**.
4	Uživatel si jednoho IdP zvolí.
5	Keycloak vytvoří žádost o autentizaci a přesměruje uživatele na vybraného IdP. Konfigurace žádosti je předem nastavená administrátorem.
6	Uživatel se přihlásí k IdP a potvrdí souhlas s požadovanými právy.
7	Při úspěšném přihlášení je uživatel přesměrován zpět na Keycloak s odpovědí, která může obsahovat bezpečnostní token. Tento token používá Keycloak pro dosažení důvěry přihlášení u IdP a také pro případný sběr atributů o uživateli.

8	V tomto kroku Keycloak ověří, že odpověď od IdP je platná. Pokud ano, mohou nastat 2 stavy: • Jedná se o nového uživatele. Keycloak vytvoří nového uživatele a může požádat IdP o uživatelské atributy, pokud již nejsou obsaženy v tokenu. Jde o **federaci identit** (identity federation). • Jedná se o existujícího uživatele. Keycloak může uživatele požádat propojení identity od IdP s existujícím účtem. Jde o **propojení účtů** (account linking). Po dokončení jednoho z možných stavů je uživatel přihlášen a je mu vytvořen vlastní token pro SeP.
9	Keycloak přesměruje uživatele k SeP i s vytvořeným tokenem.
10	SeP obdrží token a povolí uživateli přístup k požadovaným službám.

Tabulka 20 - Identity Brokering

9 Aplikační server

Poslední kapitola návrhové části se týká výběru frameworku a aplikačního serveru.

Framework pomáhá vývojářům s vývojem aplikací a usnadňuje jim život. Jelikož Keycloak je programován převážně v jazyce Java, dává smysl udržet i ukázkovou aplikaci v jazyce Java. V současné době nejpoužívanější aplikační framework pro Javu je Spring [43].

Spring je **open source framework** vytvořený původně Rodem Johnsonem. Spring vzniknul jako odlehčení technologie Jakarta EE (Java EE, J2EE). Jádrem frameworku je řešení těsných závislostí v kódu. Od vzniku roku 2004 se však framework rozrostl o další projekty, které jsou v práci využity, a to zejména Spring Boot a Spring Security.

Spring Boot usnadní práci v tom, že není potřeba se zabývat nastavením samotného aplikačního serveru a tím je ušetřen čas.

Spring Security se zabývá zabezpečením aplikace, autentizací a autorizací. Zároveň je důležité, že podporuje protokol OIDC. Keycloak také pro oba moduly nabízí adaptéry. Adaptéry jsou knihovny, které mají pomáhat s integrací.

Spring Boot v základní konfiguraci nabízí tyto open source servery:

- Jetty,
- Tomcat (výchozí),
- Undertow.

Ze srovnání rychlostí vyřizování žádostí vychází nejlépe Undertow [44]. Rozdíly mezi servery jsou však velmi malé. Co se týče podílu používání, zde je jasným vítězem Tomcat [45]. S přihlédnutím na podíl na trhu je zvolen aplikační server **Tomcat.**

9.1 Virtuální privátní server

Již bylo zmíněno, že výběr VPS není významnou součástí práce, přesto je na místě vědět jaké parametry by měly splňovat. Na VPS jsou nasazeny:

- aplikační server Tomcat s webovou aplikací,

- server s Keycloak.

Hardwarové požadavky Tomcat nejsou přímo zveřejněné. Tomcat 9 vyžaduje Java Development Kit alespoň ve verzi 8. V tabulce požadavky vychází z JDK 8. Keycloak 12.0.0 má požadavky uvedené v dokumentaci. Keycloak také vyžaduje alespoň JDK 8.

Hardwarové požadavky	Keycloak 12.0.0	Tomcat 9 (JDK 8)
CPU	Nespecifikováno	Alespoň Pentium 2 266 MHz
RAM	Alespoň 512 MB	Alespoň 128 MB
Místo na disku	Alespoň 1 GB	Alespoň 245 MB

Tabulka 21 - Hardwarové požadavky softwaru

Na obou VPS poběží **operační systém Ubuntu 20.04 Server Edition**. Požadavky systému jsou uvedeny v tabulce.

Hardwarové požadavky	Ubuntu 20.04 Server Edition
CPU	Alespoň 1 GHz
RAM	Alespoň 1 GB
Místo na disku	Alespoň 2,5 GB

Tabulka 22 - Hardwarové požadavky OS

Zmíněné požadavky z obou tabulek zvládne většina poskytovatelů VPS. Důležité pro otestování funkčnosti výsledného systému je, aby minimálně VPS, na kterém poběží Keycloak měl **veřejnou IP adresu**. Jedná se o požadavek ze strany NIA.

V další kapitole je popsána samotná implementace.

10 Implementace

Implementační část obsahuje:

- **postup sestavení systému**,
- **doprogramování nezbytných modulů**.

Celý postup sestavování systému by měl být srozumitelný, opakovatelný a použitelný v praxi. Hlavním přínosem práce je **převod uživatelských dat**, která se přenáší standardem SAML v jazyce XML, **na protokol OIDC a formát JSON**. Dalším přínosem je **vytvoření systému využitelného pro OVM**.

V tabulce jsou uvedeny verze používaného softwaru.

Software	Verze
Ubuntu SE	20.04.1
Docker	19.03.12
Keycloak	12.0.0
Spring Boot	2.3.3

Tabulka 23 - Verze softwaru

10.1 VPS a doména

Na pořízených VPS běží OS Ubuntu 20.04 Focal Fossa. Oba VPS mají veřejné IP adresy, konkrétně **167.86.68.227** a **207.180.205.220**. Zároveň je zřízena doména **testnia.tk**. Pronajmutí domény .tk je na rok zdarma. Na stránce Freenom je možné si zamluvit volnou doménu a vytvořit subdomény. S tím souvisí úpravy DNS záznamů, které se projeví do 48 hodin.

IP adresa	Subdoména
167.86.68.227	is.testnia.tk
207.180.205.220	idp.testnia.tk

Tabulka 24 - Subdomény

10.2 Docker

Nasazení softwaru na servery je řešeno pomocí platformy Docker. Docker je virtualizační technologie primárně pro Linux, která ulehčuje spouštění, testování a správu webových aplikací. Z Docker Hub je možné stáhnout připravené obrazy aplikací. Důležité je, že **každá stažená/vytvořená aplikace pracuje ve stejném prostředí** (OS, knihovny a další závislosti). Navíc spuštění je rychlejší než u klasického virtuálního stroje.

Docker pracuje s pojmy:

- **image** (obraz) – předloha prostředí pro sestavení aplikace,
- **dockerfile** – textový soubor s předpisem image,
- **container** (kontejner) – samotná aplikace, která běží v odděleném prostředí.

Instalace na obou VPS probíhá pod uživatelem root. Návod na instalaci je dostupný v dokumentaci Dockeru.

Nejdříve jsou aktualizovány dostupné balíčky na Ubuntu.

```
apt update && apt upgrade -y
```

Doinstalovány jsou potřebné aplikace.

```
apt install apt-transport-https ca-certificates curl \
gnupg-agent software-properties-common
```

Do systému je přidán oficiální Docker GPG klíč.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-
key add -
```

Dalším příkazem je přidán Docker repozitář.

```
add-apt-repository "deb [arch=amd64] \
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable"
```

V tuto chvíli je možné doinstalovat samotnou aplikaci.

```
apt update
apt install docker-ce docker-ce-cli containerd.io
```

Nyní je aplikace Docker připravena k práci, avšak pro lepší správu kontejnerů je doinstalována utilita **docker-compose**.

```
curl -L
"https://github.com/docker/compose/releases/download/1.26.2/do
cker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

Výsledné aplikace jsou publikovány ve veřejných repozitářích:

- **keycl0akn1a/is-nia** – webová aplikace s informačním systémem,
- **keycl0akn1a/idp-nia** – upravená verze Keycloak.

10.3 Keycloak

10.3.1 První spuštění

Pro práci je použita aktuálně vyvíjená verze Keycloak dostupná na GitHubu. Oficiální Keycloak repozitář je naklonován do repozitáře osobního, který je používán pro tvorbu Docker image. Pro vytvoření vlastní image je stažen repozitář keycloak-containers, protože bez něj není možné image vytvořit (důležitá je složka **server**).

```
git clone https://github.com/keycloak/keycloak-containers.git
```

Podrobný postup sestavení vlastní image Keycloak z GitHubu je dostupný zde.

Úprava souboru *Dockerfile* ve složce keycloak-container/server (dostupný zde), spočívá ve změně instrukcí ARG.

```
FROM registry.access.redhat.com/ubi8-minimal
…
ARG GIT_REPO=CarloS-JcU/keycloak
ARG GIT_BRANCH=master
…
```

Dockerfile

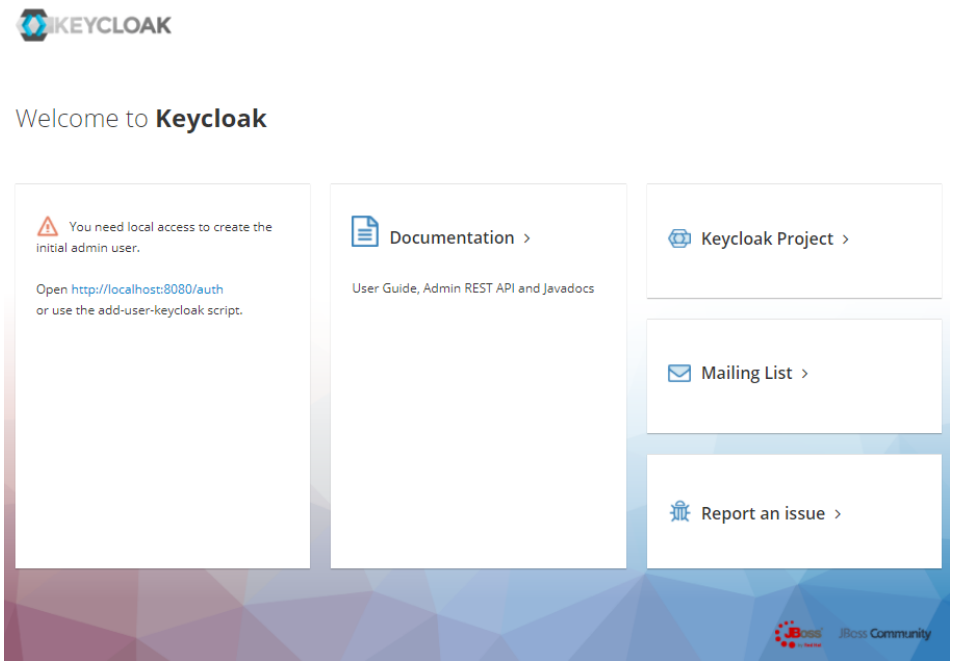
Image je sestavena pomocí příkazu *build*. Operátor *-t* slouží k pojmenování výsledné image a případně označení verze. Sestavení může trvat přes 20 minut. **Verze se mohou měnit**. Pomocí příkazu *push* je image nahraná na Docker Hub.

```
docker build -t keycl0akn1a/idp-nia:v00 .
docker push keycl0akn1a/idp-nia:v00
```

V tuto chvíli je možné vytvořenou image Keycloak spustit na serveru. V příkazu jsou použity operátory *-d* a *-p*. Operátor *-d* spustí kontejner v pozadí a operátor *-p* umožňuje namapování portů mezi hostitelem a kontejnerem. První port je hostitele, druhý pak kontejneru.

```
docker run -d -p 80:8080 keycl0akn1a/idp-nia
```

Na adrese idp.testnia.tk je zobrazena tato stránka, která informuje o nutnosti vytvoření administrátorského účtu.



Obrázek 22 - Keycloak bez admina

10.3.2 Databáze

Před veškerými změnami je doporučeno změnit výchozí relační databázi H2 za jinou. Využita je relační databáze **Postgres**. Důležité je připomenout, že v každém kontejneru by měla běžet pouze 1 aplikace. Ke spravování 2 kontejnerů je určen docker-compose. Vytvořen je soubor

docker-compose.yml s následující konfigurací. **Je důrazně doporučeno specifikovat verze používaných image**.

```
version: '3.8'

volumes:
  postgres_data:
      driver: local

services:
  postgres:
      image: postgres:12.4
      volumes:
        - postgres_data:/var/lib/postgresql/data
      environment:
        POSTGRES_DB: keycloak
        POSTGRES_USER: <ADMIN>
        POSTGRES_PASSWORD: <HESLO>

  keycloak:
      image: keycl0akn1a/idp-nia:v00
      environment:
        DB_VENDOR: POSTGRES
        DB_ADDR: postgres
        DB_DATABASE: keycloak
        DB_USER: <ADMIN>
        DB_PASSWORD: <HESLO>

      ports:
        - 80:8080
      depends_on:
        - postgres
```

docker-compose.yml

V tabulce jsou popsány jednotlivé elementy.

Elementy (pojmenování)	Popis
volumes (slovník)	Svazky, které spravuje Docker.
driver: local (slovník)	Svazky jsou dostupné na Docker hostiteli.
services	Specifikované jednotlivé služby.
image	Jaká image má být spuštěna.
environment	Specifikace proměnných. Pro Postgres jsou proměnné dostupné zde.
• POSTGRES_DB, • POSTGRES_USER, • POSTGRES_PASSWORD, • DB_VENDOR, • DB_ADDR, • DB_DATABASE, • DB_USER, • DB_PASSWORD,	• Název databáze, • administrátor databáze, • heslo administrátora k databázi, • poskytovatel databáze, • pojmenování databáze (volitelné), • název databáze, • administrátor databáze, • heslo administrátora k databázi.
ports (seznam)	Namapování portů hostitele a kontejneru.
depends_on	Závislosti spuštění. Nejdříve se spustí služba postgres a až poté keycloak.

Tabulka 25 - Docker-compose.yml

Spustit kontejnery je možné ze složky, kde se nachází soubor *docker-compose.yml* pomocí tohoto příkazu.

```
docker-compose up
```

10.3.3 Administrátor

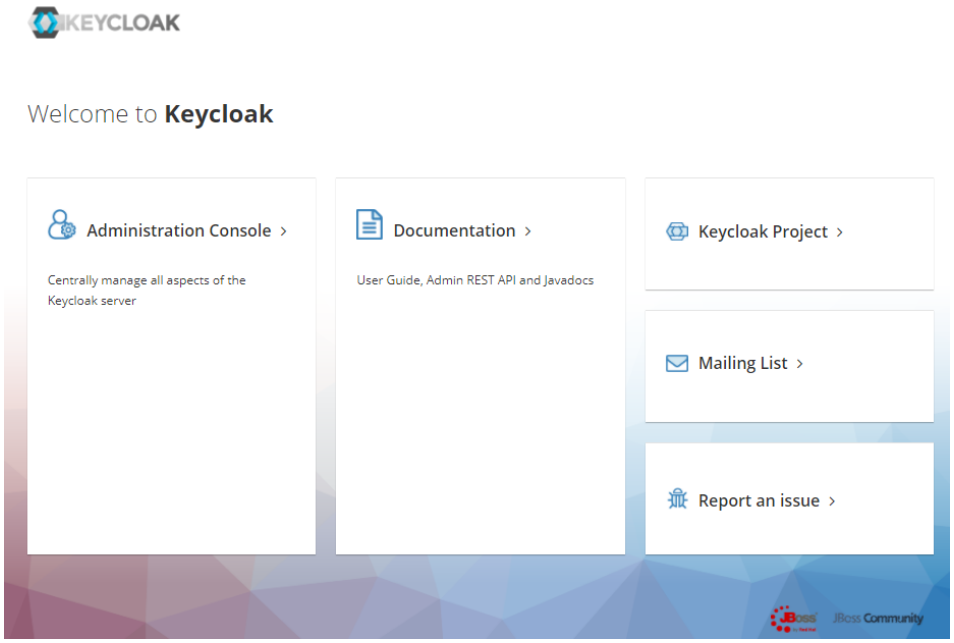
Vytvoření administrátorského účtu v Keycloak je možné v běžícím kontejneru nebo elegantněji úpravou *docker-compose.yml*. Do *environment* ve službě *keycloak* jsou přidány klíče KEYCLOAK_USER a KEYCLOAK_PASSWORD.

```
services:
  ..
  keycloak:
```

```
..
environment:
  ..
  KEYCLOAK_USER: <ADMIN>
  KEYCLOAK_PASSWORD: <HESLO>
```

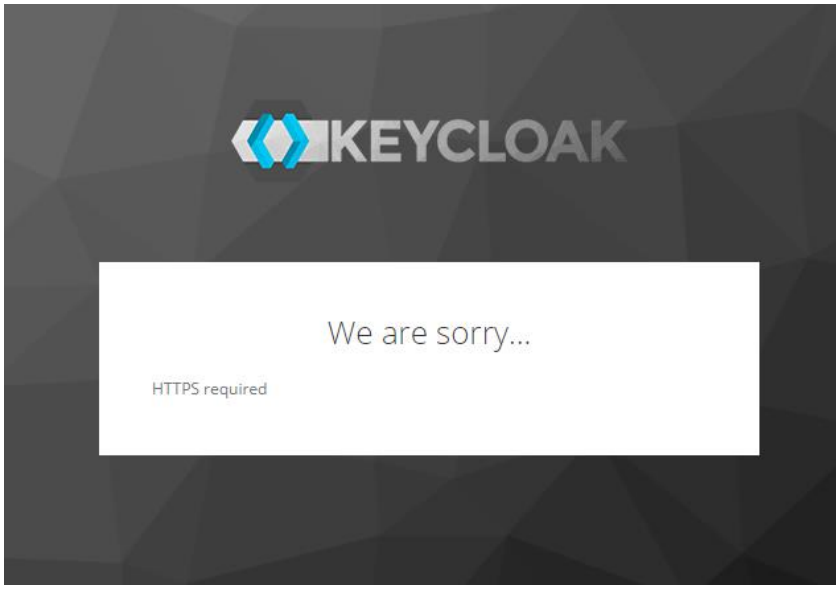
docker-compose.yml

V levém sloupci je vidět, že přihlášení administrátora je dostupné.



Obrázek 23 - Keycloak s adminem

Při pokusu o vstup do administrátorského prostředí se objeví chybová zpráva. Důvodem je nezabezpečený protokol HTTP.



Obrázek 24 - Keycloak HTTPS

10.3.4 SSL/TLS certifikáty

Pro navázání komunikace pomocí protokolu HTTPS je namapován interní port 8443 kontejneru na standardní port 443 hostitele.

```
services:
  ..
  keycloak:
      ..
      ports:
        - 80:8080
        - 443:8443
```

docker-compose.yml

Certifikáty pro správnou práci s Keycloak jsou obstarány od certifikační autority Let's Encrypt. Ta vydává certifikáty pomocí programu Certbot. Před instalací programu Certbot je doinstalován balíčkovací systém snap.

```
apt install snapd
snap install --classic certbot
```

Následovat může restart systému. Certbot využívá port 80 pro navázání spojení, proto je na místě povolit ho ve firewallu, pokud je zakázaný.

```
ufw allow 80
```

Při pokusu o spuštění Certbotu se vyskytla chyba, že port 80 na dané adrese je používaný. Nezbývá než zastavit kontejner v Dockeru.

```
docker stop <NAZEV_KONTEJNERU>
```

Nyní by výdej pomocí následujícího příkazu měl proběhnout bez problémů.

```
certbot certonly --standalone --preferred-challenges http -d
idp.testnia.tk
```

Vygenerovány jsou soubory **cert.pem**, **chain.pem**, **fullchain.pem** a **privkey.pem**, které jsou uloženy do složky /etc/letsencrypt/live/<ADRESA_WEBU>/. Na GitHubu je zmínka pouze o formátech **.crt** a **.key**. Převod na jiný formát vyřeší sám Docker. Před samotnou úpravou *docker-compose.yml* je třeba několik úprav. Certifikát (**fullchain.pem**) i privátní klíč (**privkey.pem**) jsou zkopírovány do složky ve které se nachází *docker-compose.yml*. Navíc jsou změněna práva pro čtení.

```
cp /etc/letsencrypt/live/<ADRESA_WEBU>/fullchain.pem /<CESTA
K docker-compose.yml>/
cp /etc/letsencrypt/live/<ADRESA_WEBU>/privkey.pem /<CESTA
K docker-compose.yml>/
chmod 644 fullchain.pem privkey.pem
```

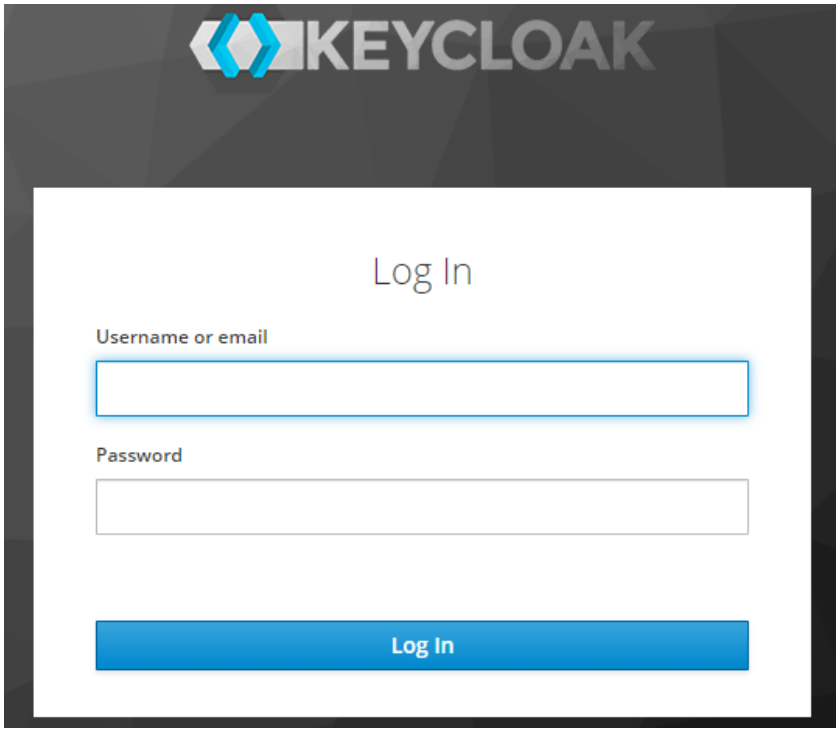
Certifikát i privátní klíč je možné připojit v *docker-compose.yml* ve slovníku *volumes* ve službě *keycloak*.

```
  keycloak:
      ..
      volumes:
        - ./fullchain.pem:/etc/x509/https/tls.crt
        - ./privkey.pem:/etc/x509/https/tls.key
```

docker-compose.yml

Po spuštění je dostupné přihlášení pro administrátora.

```
docker-compose up
```

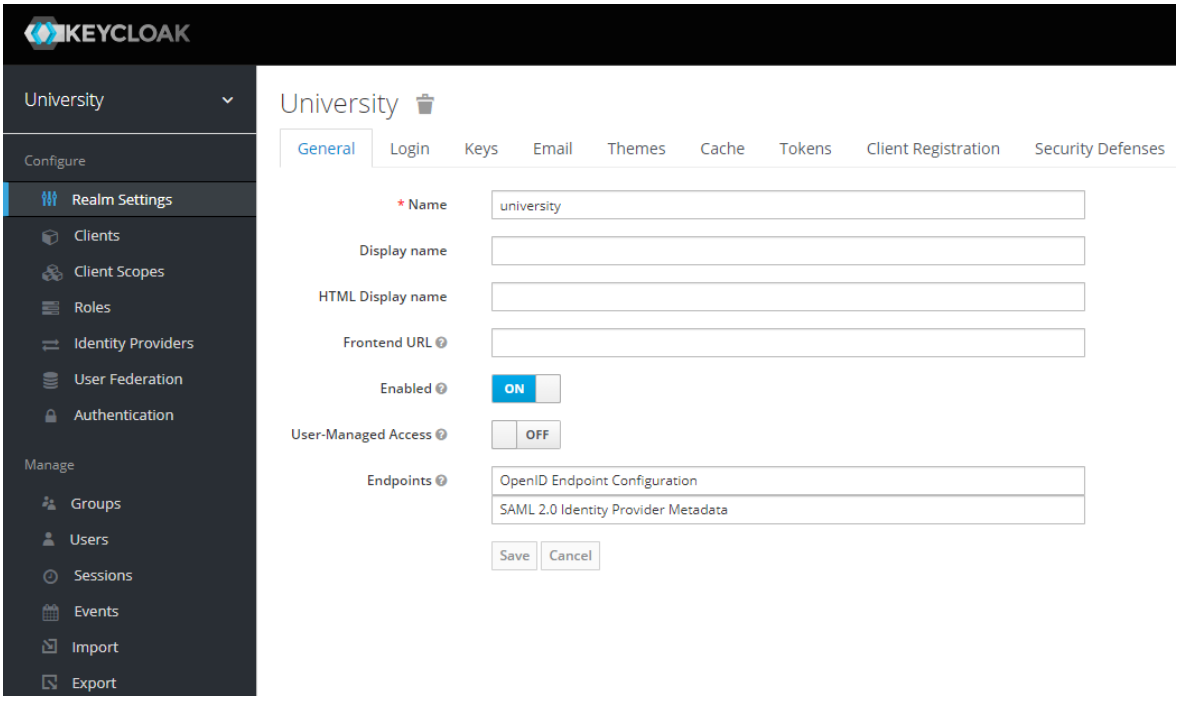


Obrázek 25 - Keycloak login

Následuje nastavení Keycloak ve webovém prostředí.

10.3.5 Realm

Aby bylo přihlašování přes IdP dostupné, musí být vytvořen realm. Po přihlášení administrátora je možné ho přidat pomocí tlačítka *Add realm*. Název realmu je *university*. Po kliknutí na tlačítko *Create* se správce dostane do základního nastavení nově vytvořeného realmu.



Obrázek 26 - Realm

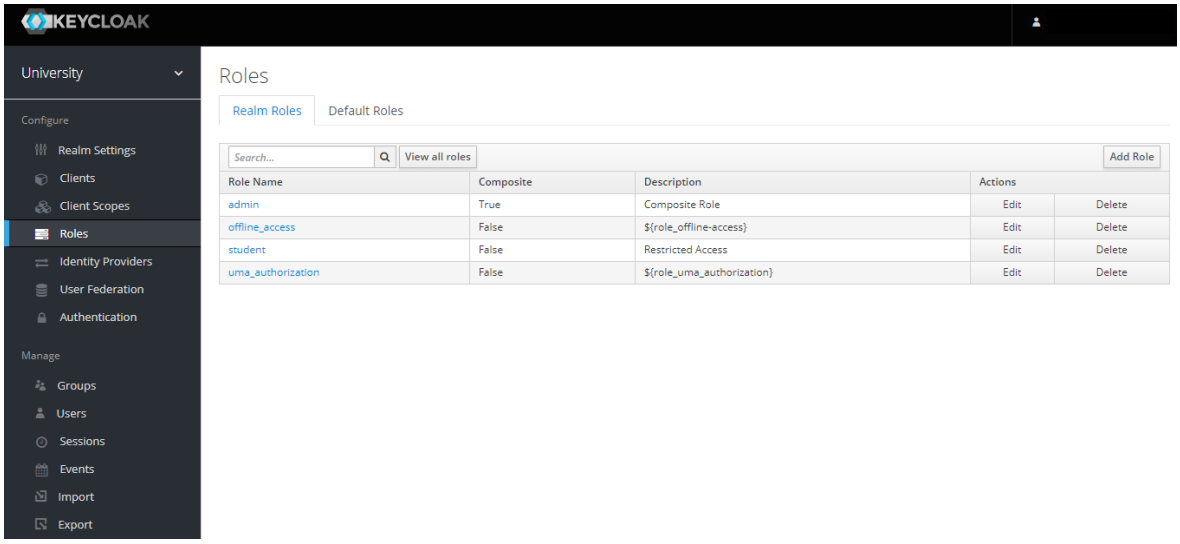
V nastavení realmu jsou vidět adresy endpointů. Pro tuto práci jsou endpointy OIDC zde a SAML zde. Ty mohou využít případní vývojáři, kteří by se chtěli napojit na vytvořený realm.

10.3.6 Role

Po vytvoření realmu jsou vytvořeny role, které reprezentují skupiny přistupujících uživatelů. Základní role jsou:

- **student**,
- správce (**admin**).

Kliknutím na *Roles* v levém panelu je možné se dostat k vytvoření rolí. Tlačítkem *Add Role* jsou přidány nové. Správce má administrativní přístup, ale zároveň by měl mít studentský přístup. Správce je *Composite Role*, tedy složená role. Tím je zajištěno, že bude mít přístup ke studentským i správcovským stránkám.

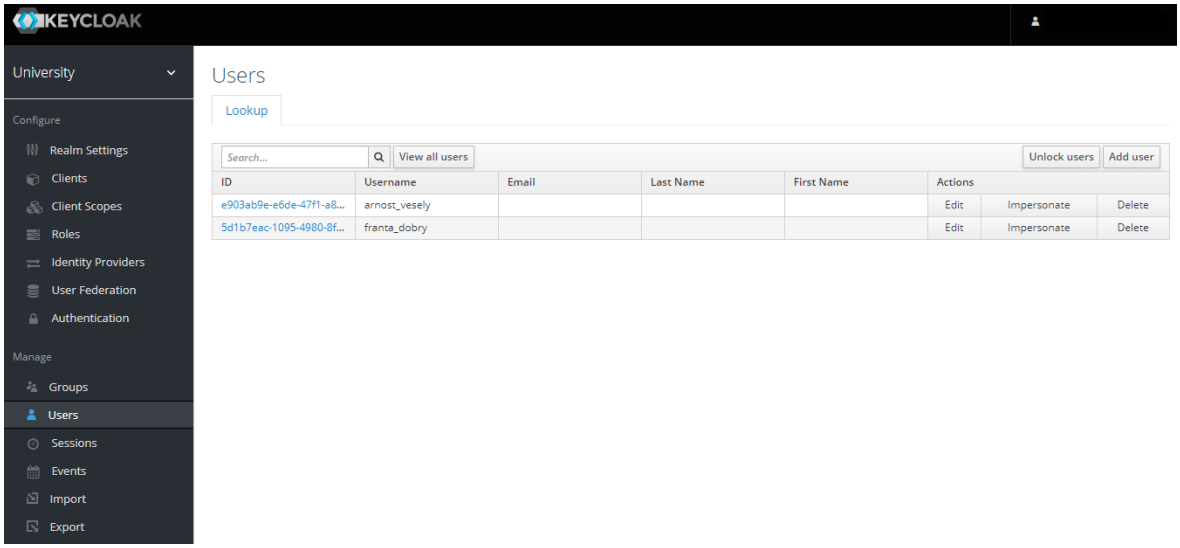


Obrázek 27 - Role

10.3.7 Uživatelé

Základní uživatelé jsou vytvořeni po kliknutí na tlačítko *Users – Add user*. Jediná požadovaná položka je *Username*. Vytvořeni jsou dva uživatelé:

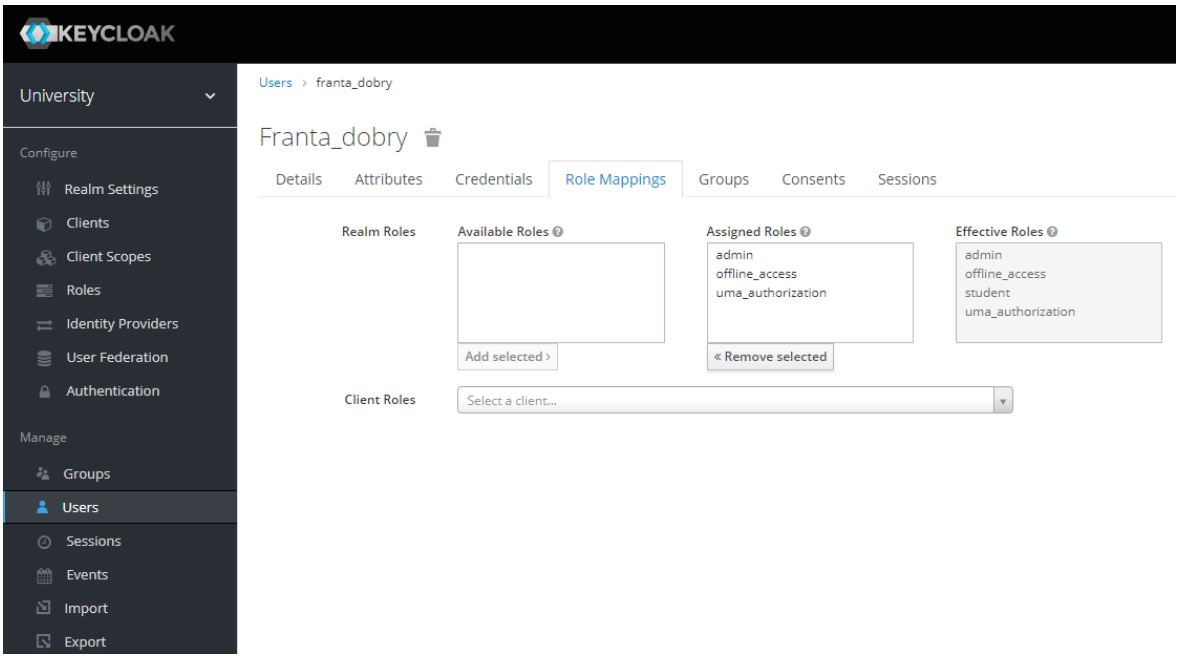
- student - *arnost_vesely*,
- správce - *franta_dobry*.



Obrázek 28 - Users

Oběma jsou změněna hesla kliknutím na ID a záložku *Credentials*.

Nastaveny jsou také role uživatelů v záložce *Role Mappings*. Správce díky složené roli zastává obě vytvořené role.

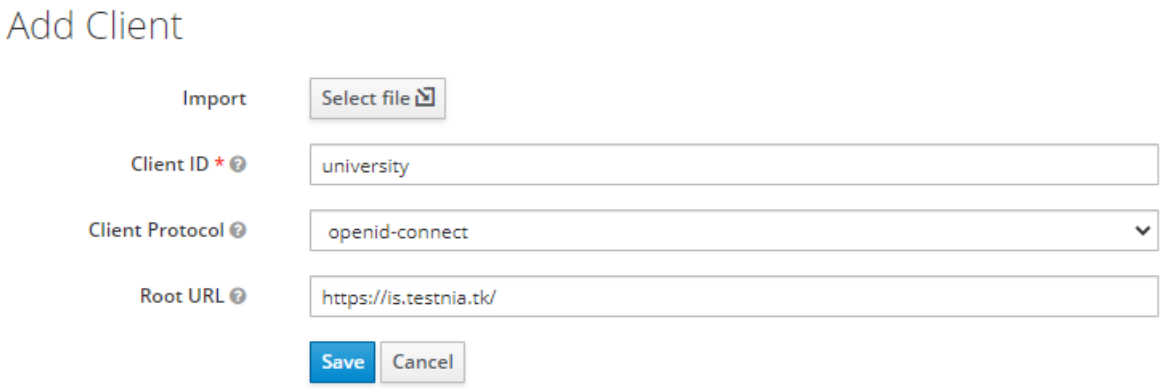


Obrázek 29 - Přidělené role

Vytvořeným uživatelům jsou přidány základní hodnoty *firstname* a *lastname*. Poslouží k oslovení uživatele na vytvořených stránkách informačního systému.

10.3.8 Client

Client reprezentuje přistupující webovou aplikaci či webovou službu. Vytvořen je kliknutím na *Clients – Create*. Jediná povinná položka je *Client ID – university*. Přednastavený je *Client Protocol* (OIDC). *Root URL* (URL webové aplikace) je adresa vytvořené webové aplikace.



Obrázek 30 - Keycloak Client

Po vytvoření *Clienta* jsou k dispozici další možnosti přizpůsobení. Je na místě zaměřit se především na položku *Valid Redirect URIs*. Zde by měly být adresy, kam je možné uživatele přesměrovat po úspěšném přihlášení nebo odhlášení. Další důležitá položka je *Access Type*. Volba *public* odpovídá Authorization Code Flow.

Při sestavování a nastavování je vycházeno především z dostupných webových návodů 1, 2 a dokumentace [46; 47; 48].

Tím je dokončena základní konfigurace Keycloak. Následuje blok o implementaci aplikace, tedy jednoduchého informačního systému, který poběží na frameworku Spring.

10.4 Spring Boot

Na webu spring initializr je nakonfigurován základní balíček. Vybrány jsou možnosti:

- Project – Maven Project,
- Language – Java,
- Spring Boot – 2.3.4 (poslední stabilní verze),
- Packaging – jar,
- Java – 8,
- Dependencies (závislosti):
    - Spring Web,

- Spring Security,
- Spring Boot DevTools,
- Thymeleaf.

Vytvořený balíček je naimportován do IDE. Výchozí aplikační server je Apache Tomcat. Do *pom.xml* je vložena závislost pro adaptér na Keycloak.

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.keycloak.bom</groupId>
                <artifactId>keycloak-adapter-bom</artifactId>
                <version>11.0.2</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

pom.xml

Následně je upravena konfigurace v souboru *application.properties*.

```
keycloak.realm=<NAZEV REALMU>
keycloak.resource=<NAZEV KLIENTA>
keycloak.auth-server-url=<ADRESA IDP SERVERU>
keycloak.ssl-required=external
keycloak.public-client=ACCESS_TYPE
```

Jelikož je dále v projektu používán modul **Spring Security**, není potřeba upravovat přístup ke stránkám v *application.properties*.

10.4.1 KeycloakConfig

Jako první je vytvořena třída *KeycloakConfig*, která zajistí správné předání údajů z *application.properties*.

```
@Configuration
public class KeycloakConfig {
    @Bean
    public KeycloakSpringBootConfigResolver
keycloakConfigResolver(){
        return new KeycloakSpringBootConfigResolver();
    }
}
```

KeycloakConfig.java

10.4.2 IsController

Další třídou je *IsController*. Ta má za úkol dohlédnout na to, aby přistupující uživatelé ke zdrojům byli autentizování a autorizování.

```
@Controller
public class IsController {
    private final HttpServletRequest request;
    private final SubjectRepository subjectRepository;

    @Autowired
    public IsController(HttpServletRequest request,
SubjectRepository subjectRepository) {
        this.request = request;
        this.subjectRepository = subjectRepository;
    }
    @GetMapping(value = "/")
    public String getHome() {
        return "index";
    }
```

IsController.java

Zde je nastaveno, kdo má přístup ke stránce *student* a *admin/login*. Na stránkách jsou vypsané dostupné předměty. V případě stránky *logout* je uživatel odhlášen.

```
    @GetMapping(value = "/student")
    public String getSubject(Model model) {
        configCommonAttributes(model);
        configCustomAttributes(model);
        model.addAttribute("subjects",
subjectRepository.readAll());
        return "student";
    }
       @GetMapping(value = "/admin/login")
    public String getManager(Model model) {
        configCommonAttributes(model);
        model.addAttribute("subjects",
subjectRepository.readAll());
        return "admin";
    }
    @GetMapping(value = "/logout")
    public String logout() throws ServletException {
        request.logout();
        return "redirect:/";
    }
```

IsController.java

V posledních metodách je získán ID Token a z něho je přečteno jméno uživatele. Takto se dají získat i další původní atributy.

```
private KeycloakSecurityContext getKeycloakSecurityContext() {
        return (KeycloakSecurityContext)
request.getAttribute(KeycloakSecurityContext.class.getName());
    }
    private void configCommonAttributes(Model model) {
        model.addAttribute("name",
getKeycloakSecurityContext().getIdToken().getGivenName());
…
    }
```

IsController.java

Vytvořený atribut/claim je třeba získat novou metodou. Claim je vytvořen a nastaven tak, aby byl zasílán v ID Tokenu. Vytvořené claims jsou řešeny v kapitole 10.7.1 Namapování atributů. Claims je možné volat přes metodu *ConfigCustomAttributes*. Vytvořen je nový String s nepůvodním Claim. Do kolekce *Map<String, Object> customClaims* jsou uloženy nestandardní claims. Poté stačí najít správnou hodnotu podle klíče a přidat do *model*. Takto je možné získat i další vytvořené claims.

```
private void configCustomAttributes(Model model) {
        String age = "";
        Map<String, Object> customClaims =
getKeycloakSecurityContext().getIdToken().getOtherClaims();
    if (customClaims.containsKey("age")) {
            age = String.valueOf(customClaims.get("age"));
        }
…
model.addAttribute("age", age);
…
}
}
```

IsController.java

10.4.3 SubjectRepository

Pro vzorovou ukázku je vložen seznam dostupných předmětů. Vytvořena je kolekce, která implementuje rozhraní *Map* a do ní jsou přidány vzorové předměty. Předměty jsou seřazeny podle ID.

```
@Repository
public class SubjectRepository {

    private static Map<String, Subject> subjects = new
ConcurrentHashMap<>();

    static {
```

```
        subjects.put("UAI7000", new Subject("UAI7000", "ID
projekt", "Karel Novák"));
        subjects.put("UAI8000", new Subject("UAI8000",
"Bezpečnost", "Jaromír Bestie"));
    }

    public List<Subject> readAll() {
        List<Subject> allSubjects = new
ArrayList<>(subjects.values());
allSubjects.sort(Comparator.comparing(Subject::getId));
        return allSubjects;
    }
```

SubjectRepository.java

Správci mají možnost přidávat či odebírat předměty.

```
    public void create(Subject subject) {
        subjects.put(subject.getId(), subject);
    }
    public void delete(String id) {
        subjects.remove(id);
    }
}
```

SubjectRepository.java

10.4.4 Subject

Pro definici objektu *Subject* je vytvořená potřebná třída. Již ze třídy *SubjectRepository* je vidět, že objekt *Subject* má 3 proměnné.

```
public class Subject {

    private String id;
    private String title;
    private String professor;
```

```
    public Subject(String id, String title, String professor)
{
        this.id = id;
        this.title = title;
        this.professor = professor;
    }
Gettery a settery
```

Subject.java

10.4.5 SecurityConfig

Třída *SecurityConfig* má na starosti, kdo má a nemá přístup ke stránkám, tedy jaká role může přistupovat k jakým stránkám. Anotace *@KeycloakConfiguration* udává, že soubor obsahuje nastavení týkající se zabezpečení. Třída dědí ze třídy *KeycloakWebSecurityConfigurerAdapter*.

```
@KeycloakConfiguration
public class SecurityConfig extends
KeycloakWebSecurityConfigurerAdapter {
```

SecurityConfig.java

V metodě *configureGlobal* je Keycloak zaregistrován jako IdP. Keycloak je zodpovědný za autentizaci a autorizaci. Spring Security používá prefix *ROLE_* + název role. V tomto případě to tedy je *ROLE_admin* a *ROLE_student*.

```
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder
auth) {
        SimpleAuthorityMapper grantedAuthorityMapper = new
SimpleAuthorityMapper();
        grantedAuthorityMapper.setPrefix("ROLE_");


        KeycloakAuthenticationProvider
keycloakAuthenticationProvider =
keycloakAuthenticationProvider();
```

```
keycloakAuthenticationProvider.setGrantedAuthoritiesMapper(gra
ntedAuthorityMapper);

auth.authenticationProvider(keycloakAuthenticationProvider);
    }
```

SecurityConfig.java

Metoda *SessionAuthenticationStrategy* udává, jaká autentizační strategie je použita pro relaci. Jelikož je zvolen Authorization Code Flow, doporučené nastavení je registrovat relaci až po úspěšné autentizaci.

```
    @Bean
    @Override
    protected SessionAuthenticationStrategy
sessionAuthenticationStrategy() {
        return new RegisterSessionAuthenticationStrategy(new
SessionRegistryImpl());}
```

SecurityConfig.java

Následující metoda obstarává, jaká role má přístup, k jakým stránkám.

- /student – student, admin.
- /admin – pouze admin.
- / - všichni.

```
  @Override
    protected void configure(HttpSecurity http) throws
Exception {
        super.configure(http);
        http
                .csrf().disable()
                .authorizeRequests()
               .antMatchers("/student").hasAnyRole("student",
"admin")
                .antMatchers("/admin/*").hasRole("admin")
                .anyRequest().permitAll();}
```

SecurityConfig.java

Poslední metoda je nutný doplněk. Bez této metody je program nespustitelný. Chyba je v tom, že třída *KeycloakWebSecurityConfigurerAdapter* používá bean se stejným názvem jako knihovna Keycloak Spring Boot Adapter.

```
    @Bean
    @Override
    @ConditionalOnMissingBean(HttpSessionManager.class)
    protected HttpSessionManager httpSessionManager() {
        return new HttpSessionManager();
    }
}
```

SecurityConfig.java

Tímto je testovací aplikace dokončena. HTML stránky, CSS styly nejsou uvedeny, protože neplní důležitou roli v práci. Celý kód je možné si prohlédnout zde.

10.4.6 SSL/TLS certifikáty

Již bylo řečeno, že aplikační server je Apache Tomcat. Certbot samotný neumí tomuto serveru aplikovat certifikáty. Vydány jsou pouze jednotlivé soubory. **Certifikáty jsou vydány na serveru**.

```
certbot certonly --standalone --preferred-challenges http -d
is.testnia.tk
```

Apache Tomcat potřebuje KeyStore ve formátu p12. KeyStore je vytvořen ve složce, kam jsou vydány soubory programem Certbot. Následující příkaz zkombinuje vydaný certifikát, privátní klíč a certifikát certifikační autority a vydá soubor *keystore.p12*, který je možný importovat na server. Zadané heslo je potřeba si pamatovat.

```
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -
out keystore.p12 -name tomcat -CAfile chain.pem -caname root
```

Do vytvořené aplikace je třeba zanést informaci o certifikátu. To je možné přidáním údajů do *application.properties*.

```
server.port: 8443
security.require-ssl=true
```

```
server.ssl.key-store: <CESTA KE KEYSTORE>
server.ssl.key-store-password: <HESLO>
server.ssl.keyStoreType: PKCS12
server.ssl.keyAlias: tomcat
```

application.properties

10.4.7 Kontejner

Před samotným vytvořením kontejneru, který bude nahrán na Docker Hub, je aplikace sestavena (v IDE příkaz *Build*). Tím je vytvořen JAR soubor aplikace. Ve stejné složce, kde je vytvořený JAR, je vytvořen *Dockerfile* s následující konfigurací.

```
FROM openjdk:8-jdk-alpine
COPY uni-is-0.0.1-SNAPSHOT.jar /app.jar
CMD ["java", "-jar", "/app.jar"]
```

Dockerfile

Kontejner je sestaven příkazem níže a nahrán na Docker Hub.

```
docker build -t keycl0akn1a/is-nia:v00 .
docker push keycl0akn1a/is-nia:v00
```

10.4.8 Nasazení

Aplikace je vyvíjena lokálně, certifikát je vydán na serveru, na kterém je zaregistrována doména. Pro spuštění je využita utilita docker-compose. Na serveru je vytvořen soubor *docker-compose.yml* s následující konfigurací.

```
version: '3.8'
services:
  is:
      image: keycl0akn1a/is-nia:v00
      volumes:
        - ./keystore.p12: <CESTA KE KEYSTORE>
      ports:
```

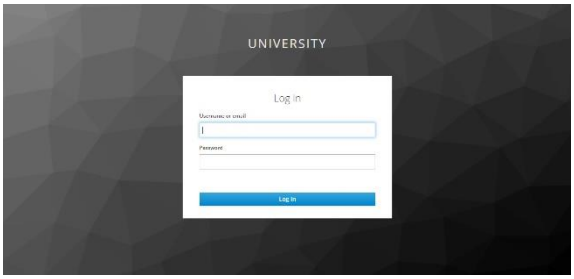
```
      - 443:8443
```

docker-compose.yml

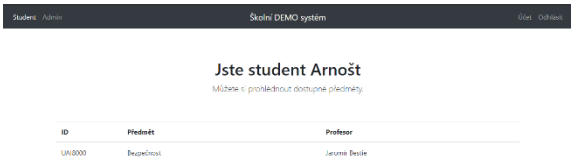
Soubor *keystore.p12* je ve stejné složce jako *docker-compose.yml* a zároveň je namapován na místo, které je specifikováno v *application.properties*.

V tuto chvíli je možné přistupovat ke stránkám is.testnia.tk protokolem HTTPS a přihlásit se pomocí Keycloak. Při vytváření webových stránek schopných komunikovat s Keycloak je vycházeno především z dostupných webových návodů 1, 2, 3 a dokumentace [49; 50; 51; 52].

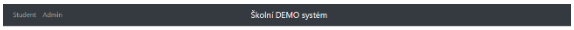
Obrázek 31 - Výchozí stránka



Obrázek 32 - Přihlášení do Keycloak

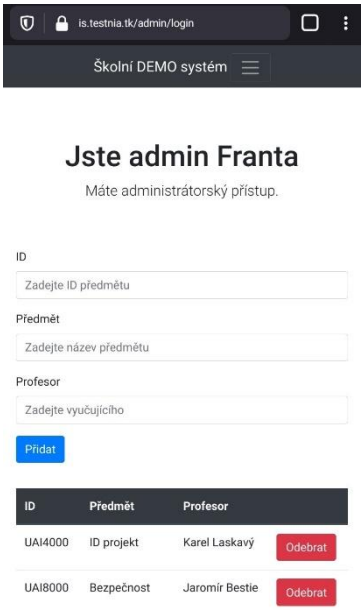


Obrázek 33 - Studentská stránka



Obrázek 34 - Blokování přístupu

Obrázek 35 - Administrátorská stránka



Obrázek 36 - Mobilní verze

Komunikace mezi ukázkovým webem a Keycloak je hotová. Následující kapitola se zabývá navázání komunikace a výměnou dat mezi Keycloak a NIA.

10.5 Keycloak a Národní identitní autorita

Keycloak sám o sobě umožňuje vytvoření vazby k poskytovateli identit, ale není možná složitější úprava požadavků. Ve webovém administrátorském rozhraní je v levém panelu dostupná možnost *Identity Providers*. Zde je možné přidat SAML v2.0 providera. Dole na stránce je možnost importovat metadata. Po naimportování testovacích NIA metadat a uložení konfigurace, jsou nastaveny základní parametry a je možné se přihlásit k NIA. Důležitý je odesílaný SAML Request.

```
<samlp:AuthnRequest
    AssertionConsumerServiceURL="https://idp.testnia.tk/auth/realms/university/broker/saml/endpoint"
    Destination="https://tnia.eidentita.cz/FPSTS/saml2/basic" ForceAuthn="false"
    ID="ID_d4c4db7a-40ec-4644-8d0a-ee5e80fb96be" IssueInstant="2020-12-03T15:13:40.690Z"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Version="2.0"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
    <saml:Issuer>https://idp.testnia.tk/auth/realms/university</saml:Issuer><samlp:NameIDPolicy
AllowCreate="true" Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/></samlp:AuthnRequest>
```

Obrázek 37 - Základní SAML Request

Jak je vidět, odchozí SAML Request je prázdný a ve webovém prostředí není mnoho možností, jak ho obohatit o požadovaná uživatelská data – <*RequestedAttributes*>. Rozšíření požadavku

SAML Request je třeba doimplementovat a s tím souvisí i implementace nového IdP v Keycloak, který posílá obohacený SAML Request.

10.5.1 Identity Provider

Vytvořený IdP je dostupný v nabídce *Identity Providers – Add Providers – Social*. Vytvořeny jsou **JAVA soubory** v nové složce ***nia*** s názvy (dostupné zde):

- ***NiaIdentityProvider***,
- ***NiaIdentityProviderConfig***,
- ***NiaIdentityProviderFactory***.

Dalším souborem je webová stránka, která obsahuje základní možnosti nastavení nového providera. Na začátek stačí zkopírovat a upravit stránku (odebrány některé položky) se základním SAML Providerem. Cesta k vytvořenému souboru ***realm-identity-provider-nia.html*** je zde.

Aby byl IdP dostupný v nabídce, je třeba upravit soubor ***org.keycloak.broker.social.SocialIdentityProviderFactory***, ke kterému vede tato cesta a připsat řádku o novém IdP.

```
org.keycloak.social.nia.NiaIdentityProviderFactory
```

org.keycloak.broker.social.SocialIdentityProviderFactory

10.5.2 NiaIdentityProviderFactory

Třída *NiaIdentityProviderFactory* dědí ze třídy *AbstractIdentityProviderFactory* a implementuje *SocialIdentityProviderFactory*.

```
package org.keycloak.social.nia;


public class NiaIdentityProviderFactory extends
AbstractIdentityProviderFactory<NiaIdentityProvider>
        implements
SocialIdentityProviderFactory<NiaIdentityProvider> {…}
```

NiaIdentityProviderFactory.java

Třída obsahuje 2 statické proměnné typu String, které jsou používány při vytváření IdP:

- *NIA_PROVIDER_ID* – doplnění názvu do automaticky vytvářeného endpointu,
- *NIA_PROVIDER_NAME* – zobrazované jméno v nabídce IdP.

Další proměnnou je pole String, díky kterému se zařadí provider do nabídky.

Poslední proměnnou je *destinationValidator* typu DestinationValidator, která je nezbytná pro vytvoření providera.

```
    public static final String NIA_PROVIDER_ID = "nia";
    public static final String NIA_PROVIDER_NAME = "NIA";
    public static final String[] COMPATIBLE_PROVIDER = new
String[]{NIA_PROVIDER_ID};
    private DestinationValidator destinationValidator;
```

NiaIdentityProviderFactory.java

Třída obsahuje přepsané metody:

- standardní gettery,
- *create* – vrací objekt typu *NiaIdentityProvider* s nastavením ve třídě *NiaIdentityProviderConfig*.

```
@Override
    public String getName() {
        return NIA_PROVIDER_NAME;}
    @Override
    public String getId() {
        return NIA_PROVIDER_ID;}
    @Override
    public NiaIdentityProvider create(KeycloakSession session,
IdentityProviderModel model) {
        return new NiaIdentityProvider(session, new
NiaIdentityProviderConfig(model),
                destinationValidator);}
```

NiaIdentityProviderFactory.java

10.5.3 NiaIdentityProviderConfig

Následuje vytvořená třída *NiaIdentityProviderConfig*, která dědí ze třídy *IdentityProviderModel*.

```
package org.keycloak.social.nia;
public class NiaIdentityProviderConfig extends
IdentityProviderModel {…}
```

NiaIdentityProviderConfig.java

Třída obsahuje 2 konstruktory:

- s parametrem *IdentityProviderModel*, který má základní obecné nastavení IdP,
- bez parametrový.

V obou konstruktorech je použit příkaz *super*, který volá původní metodu z předka. Oba konstruktory volají metodu *initialize()*, kde jsou nastaveny základní parametry komunikace s NIA a to:

- HTTP Post Binding – data jsou zasílány v HTTP hlavičce požadavku,
- SSO URL – adresa na kterou je uživatel přesměrován.

```
    private void initialize() {
        setPostBindingAuthnRequest(true);
        setPostBindingResponse(true);
        setPostBindingLogout(true);
setSingleSignOnServiceUrl("https://tnia.eidentita.cz/FPSTS/sam
l2/basic");
}
```

NiaIdentityProviderConfig.java

Zbylé metody jsou přebrané ze třídy *SAMLIdentityProviderConfig* slouží především k nastavení IdP z webového rozhraní.

10.5.4 NiaIdentityProvider

Poslední třídou je *NiaIdentityProvider*, kde probíhá základní sestavení SAML Request požadavku. Třída dědí ze třídy *AbstractIdentityProvider* a implementuje *SocialIdentityProvider*.

```
public class NiaIdentityProvider extends
AbstractIdentityProvider<NiaIdentityProviderConfig> implements
SocialIdentityProvider<NiaIdentityProviderConfig> {…}
```

NiaIdentityProvider.java

Stěžejní metodou je *performLogin*, která se stará o vytvoření SAML Requestu. V metodě je vytvářen objekt typu SAML2AuthnRequestBuilder, který skládá jednotlivé XML elementy. Volaná metoda ***addExtension*** má za úkol rozšiřovat SAML Request o element <Extensions>.

```
@Override
    public Response performLogin(AuthenticationRequest
request) {
…
SAML2AuthnRequestBuilder authnRequestBuilder = new
SAML2AuthnRequestBuilder()
.addExtension(extension)
…}
```

NiaIdentityProvider.java

Pro vytvoření požadovaných XML elementů jsou založeny třídy, jejichž instance objektu, lze volat z metody *addExtension()*.

10.5.5 SAML Request

Ukázkový SAML Request, který NIA přijme je dostupný zde (kapitola 8.3.1). Metoda *addExtension* pracuje s parametrem *(NodeGenerator extension)*. Všechny instance objektů, které jsou volány, tak musí implementovat interface *NodeGenerator*.

```
public class <NÁZEV_TŘÍDY> implements
SamlProtocolExtensionsAwareBuilder.NodeGenerator {…}
```

Vytvořené třídy jsou:

- *NiaSPType*,
- *NiaWriter*,
- *NiaCustomAttributes*,
- *NiaCustomAttribute*.

10.5.6 NiaSPType

Třída ***NiaSPType*** slouží k doplnění XML Namespace. Ve třídě jsou definovány statické proměnné typu String a objekt typu NameIDType.

```
    public static final String PREFIX = "samlp";
    public static final String URI =
"urn:oasis:names:tc:SAML:2.0:protocol";
    public static final String NS_PREFIX = "eidas";
    public static final String SAML_EXTENSIONS =
"http://eidas.europa.eu/saml-extensions";
    public static final NameIDType nameidtype = new
NameIDType();


    public static final String ELEMENT = "eidas:SPType";
    public static final String ATTRIBUTE_NAME = "public";
```

NiaSPType.java

Metoda *write* dopisuje XML Namespace k tagu <Extensions>.

```
    @Override
    public void write(XMLStreamWriter writer) throws
ProcessingException {
        StaxUtil.writeNameSpace(writer, NS_PREFIX,
SAML_EXTENSIONS);
    }
```

NiaSPType.java

Poté je vytvořena instance objektu *NiaWriter*. Objekt volá metodu *writeSptype*, která zapisuje tag <eidas:SPType>. Instanci *nameidtype* je nastavena hodnota pomocí metody *setValue* na proměnnou *ATTRIBUTE_NAME*. Metoda *writeSptype* pracuje s parametry typu NameIDType a QName. Nový QName má hodnotu proměnné *ELEMENT*. Metoda *flush* vyprazdňuje buffer zapisovače.

```
        NiaWriter niaWriter = new NiaWriter(writer);
        nameidtype.setValue(ATTRIBUTE_NAME);
        niaWriter.writeSptype(nameidtype, new QName(ELEMENT));
        StaxUtil.flush(writer);
```

NiaSPType.java

10.5.7 NiaWriter

Třída ***NiaWriter*** dědí ze třídy *BaseWriter* a v konstruktoru volá předka typu XMLStreamWriter.

```
public class NiaWriter extends BaseWriter {
       public NiaWriter(XMLStreamWriter writer) {
        super(writer);
    }
}
```

NiaWriter.java

Metoda *writeSptype* volá metodu *write* z předka *BaseWriter* a předává jí parametry.

```
   public void writeSptype(NameIDType nameIDType, QName tag)
throws ProcessingException {
        write(nameIDType, tag);
```

NiaSPType.java

Tímto jsou doplněny XML Namespace do tagu <Extensions> a vytvořen nový tag <eidas:SPType> s textem *public*. Následují třídy, které vytváří elementy <eidas:RequestedAttributes> a <eidas:RequestedAttribute>.

10.5.8 NiaCustomAttributes

Třída ***NiaCustomAttributes*** vytváří statické proměnné typu String.

```
    public static final String ELEMENT = "eidas";
    public static final String REQUESTED =
"RequestedAttributes";
    public static final String TRUE = "true";
```

NiaCustomAttributes.java

Metoda *write* vytváří počáteční element a naplňuje ho vytvořenými řetězci.

```
@Override
    public void write(XMLStreamWriter writer) throws
ProcessingException {
        StaxUtil.writeStartElement(writer, ELEMENT, REQUESTED,
"");
```

NiaCustomAttributes.java

Následně vytváří nové instance objektu *NiaCustomAttribute*, které vyžadují parametry typu String. **Právě v těchto instancích jsou specifikovány požadované atributy**. Namísto *URL_ADRESY* je třeba zadat URL adresy atributů. Ty je možné získat zde. Následuje uzavírací tag a vyprázdnění bufferu zapisovače.

```
        NiaCustomAttribute nia = new
NiaCustomAttribute("URL_ADRESA", TRUE);
        nia.write(writer);
…
        StaxUtil.writeEndElement(writer);
        StaxUtil.flush(writer);
}
```

NiaCustomAttributes.java

10.5.9 NiaCustomAttribute

Třída ***NiaCustomAttribute*** zavádí statické proměnné typu String a privátní statické proměnné, které jsou použity v konstruktoru

```
    public static final String NS_PREFIX = "eidas";
    public static final String KEY_INFO_ELEMENT_NAME =
"RequestedAttribute";
    public static final String KEY_ID_ATTRIBUTE_NAME = "Name";
    public static final String NAME_FORMAT = "NameFormat";
    public static final String NS_URI =
"urn:oasis:names:tc:SAML:2.0:attrname-format:uri";
    public static final String KEY_REQUIRED = "isRequired";

    private final String keyId;
    private final String required;
```

NiaCustomAttribute.java

V konstruktoru probíhá inicializace privátních proměnných.

```
    public NiaCustomAttribute(String keyId, String required) {
        this.keyId = keyId;
        this.required = required;
    }
```

NiaCustomAttribute.java

Následuje opět metoda *write*, která přidává počáteční element <eidas:RequestedAttribute>. Pak přidá požadované atributy, ukončovací element a vyprázdní buffer zapisovače.

```
   @Override
    public void write(XMLStreamWriter writer) throws
ProcessingException {
        StaxUtil.writeStartElement(writer, NS_PREFIX,
KEY_INFO_ELEMENT_NAME, NS_URI);
        StaxUtil.writeAttribute(writer, NAME_FORMAT, NS_URI);
        if (this.keyId != null) {
            StaxUtil.writeAttribute(writer,
KEY_ID_ATTRIBUTE_NAME, this.keyId);
        }
        if (this.required != null) {
            StaxUtil.writeAttribute(writer, KEY_REQUIRED,
this.required); }
```

```
        StaxUtil.writeEndElement(writer);
        StaxUtil.flush(writer); }
```

NiaCustomAttribute

Přidání požadované LoA je řešeno v kapitole 10.7 Přidání Identity Providera.

Rozšířený SAML Request v testovacím prostředí vypadá následovně.

```
<samlp:AuthnRequest
    AssertionConsumerServiceURL="https://idp.testnia.tk/auth/realms/university/broker/nia/endpoint"
    Destination="https://tnia.eidentita.cz/FPSTS/saml12/basic" ForceAuthn="false"
    ID="ID_76e7a4ab-75a4-4ce9-bcb1-064f443ce0b8" IssueInstant="2020-11-30T11:15:02.786Z"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Version="2.0"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
    <saml:Issuer>https://idp.testnia.tk/auth/realms/university</saml:Issuer>
    <samlp:Extensions xmlns:eidas="http://eidas.europa.eu/saml-extensions">
        <eidas:SPType>public</eidas:SPType>
        <eidas:RequestedAttributes><eidas:RequestedAttribute
Name="http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/><eidas:RequestedAttribute
Name="http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/><eidas:RequestedAttribute
Name="http://eidas.europa.eu/attributes/naturalperson/DateOfBirth"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/><eidas:RequestedAttribute
Name="http://eidas.europa.eu/attributes/naturalperson/PlaceOfBirth"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/><eidas:RequestedAttribute
Name="http://eidas.europa.eu/attributes/naturalperson/CurrentAddress"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/><eidas:RequestedAttribute
Name="http://www.stork.gov.eu/1.0/eMail"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/><eidas:RequestedAttribute
Name="http://www.stork.gov.eu/1.0/age"
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="true"/></eidas:RequestedAttributes>
    </samlp:Extensions><samlp:NameIDPolicy AllowCreate="true" Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"/>
```

Obrázek 38 - Rozšířený SAML Request

Takto upravený SAML Request splňuje požadavky podle příručky [17]. Celý kód je možné si prohlédnout zde.

10.5.10 Možnost mapování atributů

Pro **namapování atributů**, které posílá NIA je třeba vytvořit mapper (mapovač atributů). Soubor ***NiaUserAttributeMapper.java*** je vytvořen ve složce social/nia. Hlavní úkol této třídy je zavést možnost mapování User Attribute pro vytvořeného IdP. Bez této třídy jsou dostupné pouze 3 základní typy mapování.

Třída rozšiřuje původní třídu *UserAttributeMapper* a zavádí proměnnou *MAPPER_NAME*. Metoda *getCompatibleProvider* vrací název IdP, pro kterého je mapování vytvořeno. Metoda *getId* vrací proměnnou *MAPPER_NAME*.

```
public class NiaUserAttributeMapper extends
UserAttributeMapper {
```

```
    private static final String MAPPER_NAME = "NIA-attribute-
mapper";
    @Override
    public String[] getCompatibleProviders() {
        return NiaIdentityProviderFactory.COMPATIBLE_PROVIDER;
    }
    @Override
    public String getId() {
        return MAPPER_NAME;
    }
}
```

NiaUserAttributeMapper.java

Vytvořený mapovač je zaveden do seznamu mapovačů. To je možné přidáním řádku do souboru *org.keycloak.broker.provider.IdentityProviderMapper*.

```
…
org.keycloak.social.nia.NiaUserAttributeMapper
…
```

org.keycloak.broker.provider.IdentityProviderMapper

Tímto způsobem je možné přidávat i další možnosti mapování.

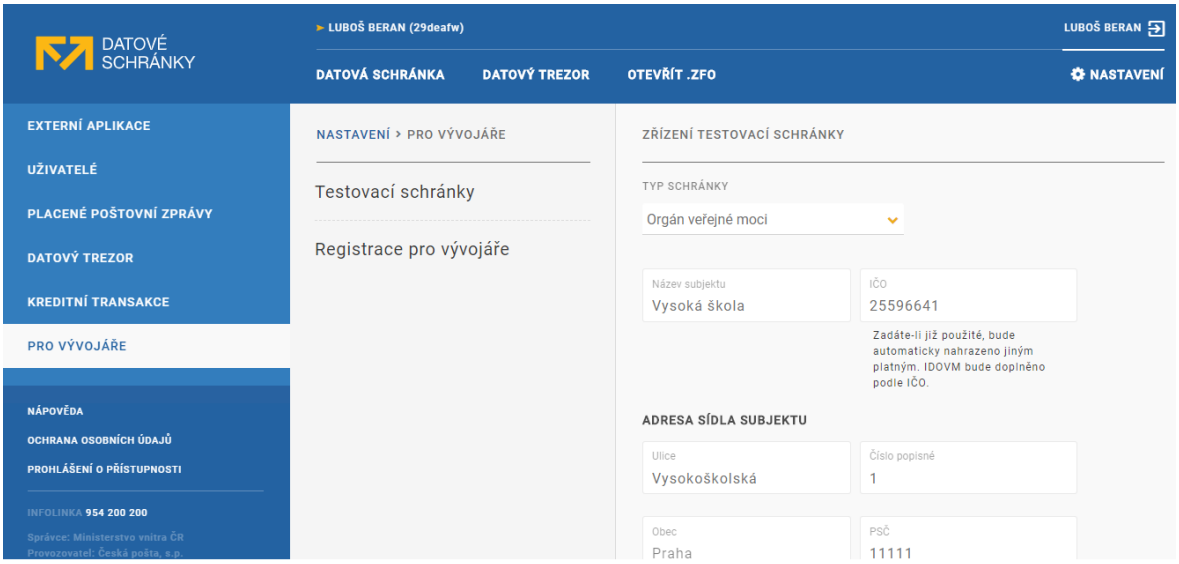
Nyní je nezbytné aktualizovat kód na GitHubu a aktualizovat image na Docker Hub. V další kapitole je řešeno napojení Keycloak a NIA.

10.6 Testovací profil

Připojení k NIA je možné po založení testovací DS a vytvoření testovacího kvalifikovaného poskytovatele (SeP).

10.6.1 Testovací datová schránka

Pro založení testovací DS, je třeba mít již běžnou DS zřízenou. Po přihlášení k DS je možné v nastavení v záložce *Pro vývojáře* vytvořit testovací DS.



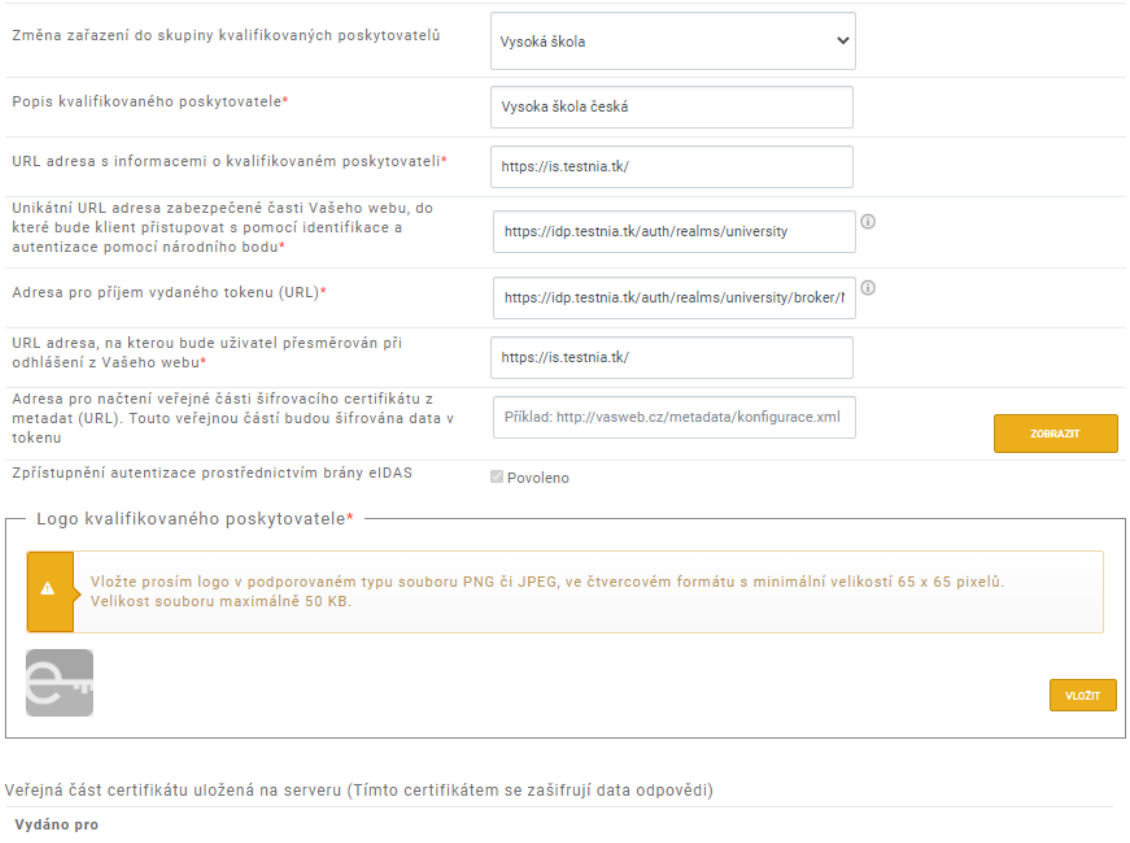
Obrázek 39 - Testovací datová schránka

Typ schránky je zvolen OVM, aby byla zachována co největší autentičnost. Registrované údaje se nepromítnou do reálného prostředí. Testovací DS je možné použít na webu czebox (testovací přihlašovací portál). Ihned po přihlášení následuje povinná změna hesla. Poté je schránka zpřístupněna.

10.6.2 Testovací kvalifikovaný poskytovatel

Pro založení testovacího SeP stačí na testovací portále NIA kliknout na *Kvalifikovaný poskytovatel online služeb* a přihlásit se k testovací DS.

Po odsouhlasení předání údajů je provedena *Registrace organizace*. Údaje jsou převzaty z testovací DS. Poté je zpřístupněna položka *Konfigurace kvalifikovaného poskytovatele*. Na této stránce jsou zadávány základní informace o SeP a také URL adresy.



Obrázek 40 - Konfigurace SeP

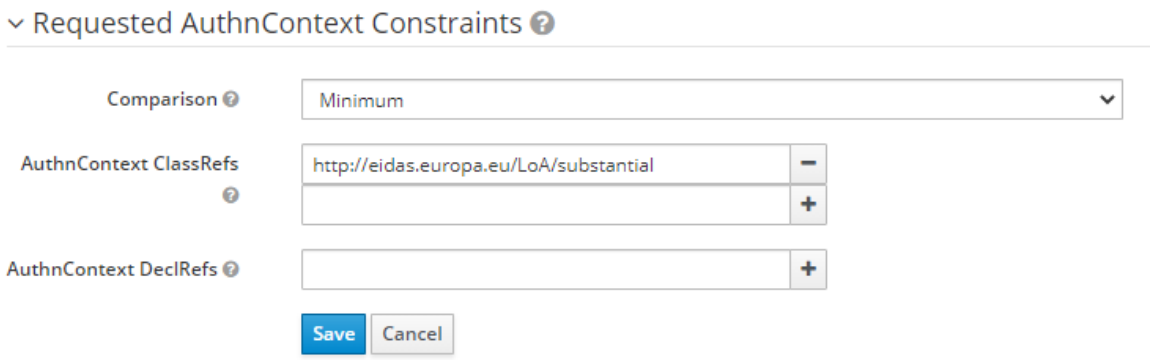
Důležité parametry jsou:

- unikátní URL adresa zabezpečené části webu – *SAML Issuer* – adresa realmu,
- adresa pro příjem vydaného tokenu (URL) – *SAML Audience* - adresa endpointu,
- veřejná část certifikátu – dostupné jsou dvě možnosti:
    - zadat adresu vystavených metadat,
    - nahrání textového souboru s certifikátem, této možnosti je využito v této práci. Veřejná část certifikátu Keycloak lze získat v *Realm Settings – SAML 2.0 IdP Metadata*.

Tímto dojde k vytvoření důvěryhodného vztahu mezi NIA a Keycloak. Následuje přidání IdP ve webovém rozhraní Keycloak.

10.7 Přidání Identity Providera

V Keycloak v administrátorském prostředí je možné přidat IdP v menu kliknutím na volbu *Identity Providers – Add provider…* Zde je možnost „NIA“. Po kliknutí je zobrazeno nastavení komunikace (stránka se bude dále s vývojem měnit). Při úpravách je dbáno na to, aby případný správce měl co nejméně práce s nastavením. Prozatím jedinou nutností je nastavit ***Requested AuthnContext Constraints*** na požadovanou hodnotu LoA.



Obrázek 41 - Requested AuthnContext Constraints

Poté stačí kliknout na tlačítko *Save*. Přistupující student z informačního systému, tak má dostupnou možnost přihlásit se a ověřit se vůči NIA.

10.7.1 Namapování atributů

Příchozí SAML atributy je třeba namapovat na OIDC claims. Tabulka přehledně zobrazuje páry atributů/claims/property. Property jsou názvy mapovačů pro claim, které používá Keycloak při komunikaci s informačním systémem. Je možné si je zobrazit v záložce *Client Scopes*.

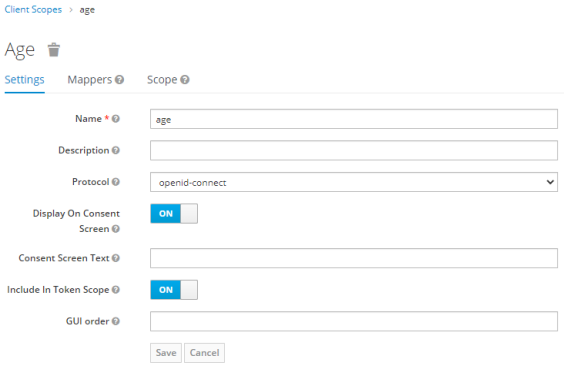
- Pokud jde o původní scope a původní claim, jedná se o položku *Property*.
- Pokud jde o vytvořený scope a vytvořený claim, jedná se o položku *User Attribute*.

SAML atributy	OIDC Claims	Keycloak Property
CurrentFamilyName	family_name	lastName
CurrentGivenName	given_name	firstName
DateOfBirth	birthdate	birthdate (User Attribute)

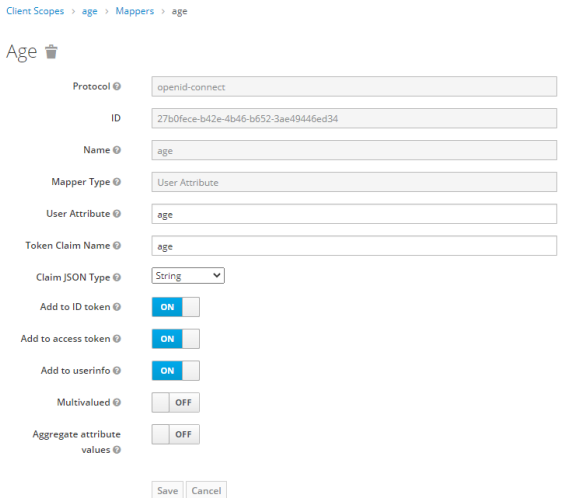
PlaceOfBirth	**placeOfBirth**	placeOfBirth (User Attribute)
CurrentAddress	**addrezz**	addrezz (User Attribute)
Email	email	email
Age	**age**	age
PhoneNumber	phone_number	phoneNumber

Tabulka 26 - Namapování atributů

Claims *placeOfBirth*, *age*, *addrezz* nejsou původní, ale vytvořené. Název *addrezz* je zvolen tak, aby nekolidoval s původním *address*. Zasílané claims je možné vytvořit pod záložkou *Client Scopes* tlačítkem *Create*. Ihned je vhodné založit mapper.

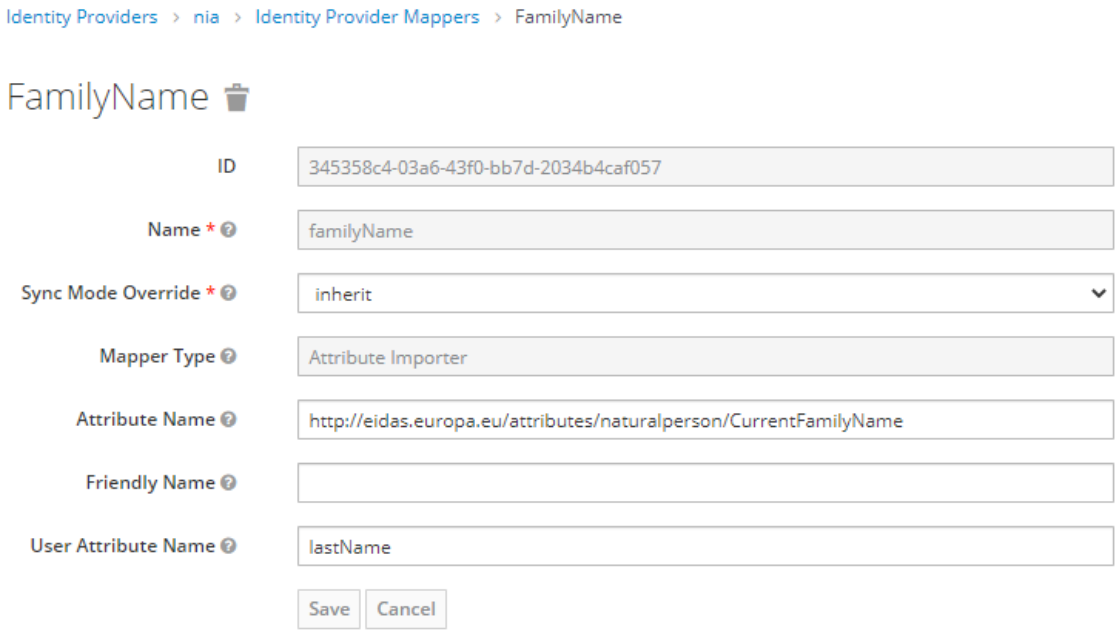


Obrázek 42 - Client Scopes



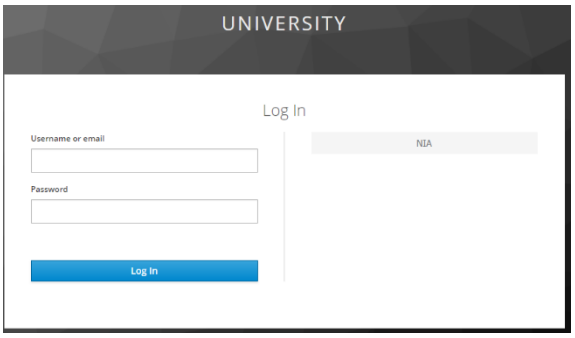
Obrázek 43 - Client Scopes - Mappers

Ve webovém prostředí v nastavení jednotlivých IdP je záložka *Mappers*. Zde probíhá namapování zasílaných atributů na claims. V položce *name* je možné vyplnit libovolný název. Jako *Mapper Type* je zvolen *Attribute Importer*. Do *Attribute Name* je třeba zadat URL SAML atributu, ty jsou dostupné zde pod *Name (ClaimType)*. Jako *User Attribute Name* je nutné vyplnit *Keycloak Property* z tabulky výše. Na obrázku je ukázka nastaveného mapovače.



Obrázek 44 - Mapper

Na obrázcích níže je vidět postup případného uživatele při přihlášení k informačnímu systému.



Obrázek 45 – Přihlášení

Obrázek 46 - Výběr testovacího profilu

Obrázek 47 - Přihlášení k NIA

Obrázek 48 - Souhlas s předáním údajů

Při pokusu o přihlášení se ovšem objeví chyba, která značí problém v parseru. Parser nezná typy, které NIA používá. Jedná se o typy:

- ***CurrentFamilyNameType***,
- ***CurrentGivenNameType***,
- ***DateOfBirthType***,
- ***PlaceOfBirthType***,
- ***CurrentAddressType***.

Pro správnou funkci je upravena třída ***SAMLAttributeValueParser***, která má na starosti parser SAML atributů.

10.7.2 Úprava parseru

Pro správné zpracování většiny typů je úprava jednoduchá. Problémem je typ ***CurrentAddressType***, kterému je věnována samostatná podkapitola. Úpravy jsou prováděny v souboru *SAMLAttributeValueParser.java*, který je možné nalézt zde. Doplněna je metoda *parse*, konkrétně část starající se o to, co se má stát, pokud je nalezen daný typ. Typy, které je možné převést na String, jsou zpracovány metodou *getElementText* ze třídy *StaxParserUtil*.

```
…
String typeValue = StaxParserUtil.getAttributeValue(type);
        if (typeValue.contains(":string")) {
            return
StaxParserUtil.getElementText(xmlEventReader);
}
…
        else if (typeValue.contains(":CurrentFamilyNameType"))
{
            return
StaxParserUtil.getElementText(xmlEventReader);
}
…
```

SAMLAttributeValueParser.java

Typ *DateOfBirthType* je zpracován podobně jako původní typ *date*. Pro něj je volána metoda *parse* ze třídy *XMLTimeUtil*, která převede datum na String.

```
else if (typeValue.contains(":date")) {
            return
XMLTimeUtil.parse(StaxParserUtil.getElementText(xmlEventReader
));)
}
else if (typeValue.contains(":DateOfBirthType")) {
```

```
            return
XMLTimeUtil.parse(StaxParserUtil.getElementText(xmlEventReader
));
}
```

SAMLAttributeValueParser.java

Tímto jsou nestandardní typy, převedeny na srozumitelný typ String. Posledním řešeným typem je *CurrentAddressType*, který je řešen v následující podkapitole.

10.7.3 CurrentAddressType

Pokud parser narazí na *CurrentAddressType* je zavolána vytvořená metoda *getAddressElement* ze třídy *StaxParserUtil*.

```
String typeValue = StaxParserUtil.getAttributeValue(type);
…
else if (typeValue.contains(":CurrentAddressType")) {
            return
StaxParserUtil.getAddressElement(xmlEventReader); }
```

Získaná data jsou nejdříve osekána a dekódována.

```
public static String getAddressElement(XMLEventReader
xmlEventReader) throws ParsingException {
        String str = null;
        try {
            str = xmlEventReader.getElementText().trim();
            byte[] valueDecoded = Base64.decode(str);
```

StaxParserUtil.java

Následuje převedení na typ String a zavolání nově vytvořené metody *currentAddressTypeParser*, která se stará o rozdělení dat.

```
  String address = new String(valueDecoded);
            str = currentAddressTypeParser(xmlEventReader,
address);
```

StaxParserUtil.java

Metoda *currentAddressTypeParser* nejprve rozdělí jednotlivé řádky a z nich poté oseká vše co je za znakem > a před znakem <. Takto osekaná data vezme a uloží do proměnné typu String.

```
String result = "";
        String[] lines = address.split("\\r?\\n");
        for (int i = 0; i < 5; i++) {
            lines[i] = lines[i].split("[>]")[1];
            lines[i] = lines[i].split("[<]")[0];
            result += lines[i] + " ";
        }
```

StaxParserUtil.java

Jednotlivé hodnoty XML elementů jsou uloženy jako jeden String. Zde je v dalším vývoji předpokládaná úprava, aby jednotlivé hodnoty bylo možné namapovat jednotlivě.

Toto je poslední úprava a **pro přistupující zájemce o studium v případě vysoké školy** nebo **zájemce o služby OVM**, **je zpřístupněna možnost důvěryhodně se identifikovat a autentizovat vůči NIA**.

Zasílaný JSON Web Token obsahuje snadno čitelné claims.


```
{
    "sub": "63af1acb-56f8-4a12-8d58-639ddabc711e",
    "placeOfBirth": "Praha 4",
    "birthdate": "1947-07-14",
    "addrezz": "10/13b Bělehradská Praha 4 14000 Praha, Nusle ",
    "name": "BOROVICE PYRENEJSKÁ",
    "preferred_username": "borovice",
    "given_name": "BOROVICE",
    "family_name": "PYRENEJSKÁ",
    "age": "73",
    "email": "borovice@borovice.com"
}
```


Obrázek 49 - JSON Web Token

10.8 Postup nasazení

Pro zpřehlednění jednotlivých kroků pro nasazení je doplněn seznam jednotlivých zjednodušených akcí.

1. Zřídit 2 servery s veřejnými IP adresami a linuxovým OS (Docker je zaměřen primárně na Linux). Obstarat domény. Kapitola 10.1 VPS a doména.
2. Nainstalovat aplikace pro práci s gitem, Dockerem a utilitu docker-compose. Kapitola 10.2 Docker.
3. Připravit si SSL/TLS certifikáty. Kapitola 10.3.4 SSL/TLS certifikáty a 10.4.6 SSL/TLS certifikáty
4. Dodat certifikáty serverům.
    a. Pro Keycloak je třeba upravit *docker-compose.yml*. Případně změnit databázi. V příloze A je uvedeno kompletní nastavení souboru *docker-compose.yml*.
    b. Pro IS je zapotřebí stáhnout kód z GitHubu a upravit soubor *application.properties*. Zároveň certifikát převést do *keystore.p12*. V příloze B je uvedeno kompletní nastavení souboru *docker-compose.yml*.
5. Nastavit Keycloak. Kapitola 10.3 Keycloak.
    a. Vytvořit Realm. Kapitola 10.3.5 Realm.
    b. Vytvořit Client Scopes a claims, ideálně s nastavením názvů na:
        - addrezz,
        - age,
        - placeOfBirth.

    Kapitola 10.7.1 Namapování atributů.
    c. Vytvořit Clienta. Kapitola 10.3.8 Client.
    d. V nastavení Client upravit zasílané Client Scopes.
    e. Vytvořit Role. Kapitola 10.3.6 Role.
    f. Založit NIA IdP. Kapitola 10.7 Přidání Identity Providera.
        - Nezapomenout nastavit požadovanou LoA dole na stránce!

g. Namapovat atributy na Claims. Kapitola 10.7.1 Namapování atributů.

6. Založit testovací DS a vyplnit Konfiguraci testovacího poskytovatele na webu testovací eIdentita. Kapitola 10.6 Testovací profil.
7. Otestovat nasazení a případně opravit nedostatky.

11 Závěr

V závěru jsou zopakovány cíle a to, zda se je podařilo splnit.

- Popsat Národní identitní autoritu a s ní spojené pojmy.
    - Ano, věnována kapitola 2 Národní identitní autorita.
- Seznámit se zákony týkající se Národní identitní autority.
    - Ano, taktéž v kapitole 2 Národní identitní autorita.
- Seznámit se zákony týkající se vysokých škol jako orgány veřejné moci.
    - Ano, díky novele zákona č. 300/2008 Sb., kdy vysoká škola je orgán veřejné moci. Rozebráno v kapitole 4 Vysoké školy.
- Provést věcný rozbor ukotvení elektronické identity v právním řádu České republiky a Evropské unie.
    - Ano, elektronická identita a nařízení eIDAS jsou rozebrány v kapitole 3 Elektronická identifikace v evropském kontextu.
- Popis existujících softwarových modulů pro implementaci služeb Národní identitní autority.
    - Ano, tomuto cíli jsou věnovány kapitoly 8 Identity & Access Management software a 9 Aplikační server, kde jsou vybrány softwarové aplikace, které je možné použít pro implementaci.
- Vytvoření studie proveditelnosti implementace služeb Národní identitní autority do informačního systému vysoké školy.
    - Ano, kapitola 10 Implementace je věnována tomu, jak sbírat data od Národní identitní autority, jak byl dopracován kód v Keycloak a jak přenášet na vytvořený jednoduchý informační systém.

Z jednotlivých bodů a podbodů vyplývá, že náplň diplomové práce se podařilo naplnit. Vytvořené řešení je dostupné na GitHubu (*Keycloak*, *informační systém*) a také na platformě Docker Hub (*Keycloak*, *informační systém*).

Systém je možné nasadit a mít tak zjednodušený start na to, jak proniknout do systému NIA. Díky práci je také otevřena cesta menším OVM využívat systém NIA. Vysoké školy mají předpřipravené řešení, které umožní rychlejší digitalizaci agendy. Data od NIA jsou převáděna do modernějšího formátu JSON a v práci je popsáno, jak je získat z Keycloak. Důležité je však myslet na to, že se jedná o testovací prostředí. Vedle dostupného řešení pomocí jazyka PHP nia.otevrenamesta.cz se jedná pravděpodobně o druhé veřejně dostupné řešení integrace NIA.

Práci je možno dále rozvíjet a také bude rozvíjena. Základním doplněním je dopsat přívětivé a popisné README na GitHub. Je nezbytné vyřešit, co se stane, pokud uživatel neodsouhlasí předání atributů. V tuto chvíli je přesměrován na endpoint Keycloak. S testovací NIA je třeba vyřešit proč nevydává telefonní číslo. Data z atributu *CurrentAddress* je třeba lépe zpracovat a namapovat na již existující claims. Možným rozšířením je nasazení v produkčním prostředí a popsat případné problémy a chyby. Velká příležitost je vytvořit mobilní aplikaci, která bude používat protokol OIDC a přijímat data od vytvořeného Keycloak. Dalším, spíše menším rozšířením, je grafické doladění možnosti přihlášení a případně vytvořit grafický kabátek ke zlepšení uživatelské přívětivosti.

Aktivity na práci nejsou u konce, protože je možnost jí stále zlepšovat a je přislíbena spolupráce při případném nasazení do produkčního prostředí.

12 Seznam literatury

[1] EVROPSKÁ UNIE. *Nařízení Evropského parlamentu a Rady (EU) č. 910/2014: o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu a o zrušení směrnice 1999/93/ES*. In: . 2014, ročník 2014, číslo 910. Dostupné také z: https://eur-lex.europa.eu/legal-content/CS/TXT/PDF/?uri=CELEX:32014R0910&from=CS

[2] MINISTERSTVO VNITRA ČESKÉ REPUBLIKY. Národní identitní autorita. *Národní architektura eGovernmentu veřejné správy ČR* [online]. [cit. 2020-07-02]. Dostupné z: https://archi.gov.cz/nap:nia

[3] OASIS. *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. 02. 2008. Dostupné také z: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

[4] *IEEE standard glossary of software engineering terminology*. New York, N.Y.: Institute of Electrical and Electronics Engineers, 1990. ISBN 15-593-7067-X.

[5] Druhy právních předpisů EU. *Evropská komise* [online]. [cit. 2020-01-03]. Dostupné z: https://ec.europa.eu/info/law/law-making-process/types-eu-law_cs

[6] EVROPSKÁ UNIE. *Nařízení Evropského parlamentu a Rady (EU) č. 2019/1157: o posílení zabezpečení průkazů totožnosti občanů Unie a povolení k pobytu vydávaných občanům Unie a jejich rodinným příslušníkům, kteří vykonávají své právo volného pohybu*. In: . 2019, ročník 2019, číslo 1157. Dostupné také z: https://eur-lex.europa.eu/legal-content/CS/TXT/PDF/?uri=CELEX:32019R1157&from=EN

[7] RAŠEK, Luděk. Druhá šance pro elektronickou občanku. Do dvou let bude muset být bezkontaktní. *Lupa.cz* [online]. [cit. 2020-07-07]. Dostupné z: https://www.lupa.cz/clanky/druha-sance-pro-elektronickou-obcanku-do-dvou-let-bude-muset-byt-bezkontaktni/

[8] Na digitalizaci vysokých škol dá MŠMT 170 mil. *Ministerstvo školství, mládeže a tělovýchovy* [online]. c2013-2020 [cit. 2020-11-28]. Dostupné z: https://www.msmt.cz/ministerstvo/novinar/na-digitalizaci-vysokych-skol-da-msmt-170-mil

[9] DZURILLA, Vladimír. *Informační koncepce České republiky: Koncepce budování eGovernmentu v ČR 2018+ a jeho IT podpory podle zák. 365/2000 Sb., o informačních systémech veřejné správy a o změně některých dalších zákonů, ve znění pozdějších předpisů*. Finální verze. 2018. Dostupné také z: https://www.mvcr.cz/soubor/vladni-program-digitalizace-ceske-republiky-2018-digitalni-cesko-informacni-koncepce-cr.aspx

[10] The Digital Economy and Society Index (DESI). *European Commission* [online]. 2020. Dostupné také z: https://ec.europa.eu/digital-single-market/en/desi

[11] Rada vlády pro informační společnost. *Ministerstvo vnitra České republiky* [online]. 2019 [cit. 2019-10-16]. Dostupné z: https://www.mvcr.cz/clanek/rada-vlady-pro-informacni-spolecnost.aspx?q=Y2hudW09Ng%3d%3d

[12] HAVLÍČEK, Karel. *Inovační strategie České republiky 2019–2030*. Praha, 2019, 28 s. Dostupné také z: https://www.vlada.cz/assets/urad-vlady/poskytovani-informaci/poskytnute-informace-na-zadost/Priloha_1_Inovacni-strategie.pdf

[13] AQUARO, Vincenzo. United Nations E-government survey 2018: Gearing E-Government to Support Transformation Towards Sustainable and Resilient Societies. *United Nations: Department of Economic and Social Affairs* [online]. s. 269 [cit. 2020-01-16]. Dostupné z: doi:978-92-1-123205-9

[14] ČESKÁ REPUBLIKA. Zákon č. 21/1992 Sb.: Zákon o bankách. In: *Sbírka zákonů*. 1992, ročník 1992, 5/1992, číslo 21. Dostupné také z: https://www.zakonyprolidi.cz/cs/1992-21

[15] EVROPSKÁ UNIE. *Prováděcí nařízení komise (EU) 2015/1502: kterým se stanoví minimální technické specifikace a postupy pro úrovně záruky prostředků pro elektronickou identifikaci podle čl. 8 odst. 3 nařízení Evropského parlamentu a Rady (EU) č. 910/2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu*. In: .

2015, ročník 2015, číslo 1502. Dostupné také z: https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=CELEX%3A32015R1502

[16] ČESKÁ REPUBLIKA. *Zákon č. 250/2017 Sb.: Zákon o elektronické identifikaci*. In: . 2017, ročník 2017, číslo 250. Dostupné také z: https://www.zakonyprolidi.cz/cs/2017-250

[17] SPRÁVA ZÁKLADNÍCH REGISTRŮ. *Příručka k využití služeb národní identitní autority pro poskytovatele služeb veřejné správy*. 1.7. 2019. Dostupné také z: https://info.eidentita.cz/download/SeP_PriruckaKvalifikovanehoPoskytovatele.pdf

[18] ČESKÁ REPUBLIKA. *Zákon č. 186/2016 Sb.: Zákon o hazardních hrách*. In: . 2016, ročník 2016, 71/2016, číslo 186. Dostupné také z: https://www.zakonyprolidi.cz/cs/2016-186

[19] Kvalifikovaný poskytovatel online služeb. *EIdentita.cz* [online]. 2019 [cit. 2019-10-31]. Dostupné z: https://www.eidentita.cz/Home/Ovm

[20] EVROPSKÁ UNIE. *Prováděcí nařízení komise (EU) 2015/1501: o rámci interoperability podle čl. 12 odst. 8 nařízení Evropského parlamentu a Rady (EU) č. 910/2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu*. In: . 2015, ročník 2015, číslo 1501. Dostupné také z: https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=CELEX%3A32015R1501

[21] O projektu IS ORG. *Úřad pro ochranu osobních údajů* [online]. [cit. 2020-07-05]. Dostupné z: https://www.uoou.cz/o-projektu-is-org/ds-1939/archiv=0

[22] ČESKÁ REPUBLIKA. *Zákon č. 111/1998 Sb.: Zákon o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách)*. In: . 1998, ročník 1998, 39/1998, číslo 111. Dostupné také z: https://www.zakonyprolidi.cz/cs/1998-111

[23] PETERKA, Jiří. Jak se přihlašuje pomocí nové elektronické občanky?. *Lupa.cz* [online]. [cit. 2019-11-09]. Dostupné z: https://www.lupa.cz/clanky/jak-se-prihlasuje-pomoci-nove-elektronicke-obcanky/

[24] ČESKÁ REPUBLIKA. Zákon č. 300/2008 Sb.: Zákon o elektronických úkonech a autorizované konverzi dokumentů. In: *Sbírka zákonů*. 2008, ročník 2008, 98/2008, číslo 300. Dostupné také z: https://www.zakonyprolidi.cz/cs/2008-300

[25] Evropská komise. *Evropská unie* [online]. [cit. 2019-12-19]. Dostupné z: https://europa.eu/european-union/about-eu/institutions-bodies/european-commission_cs

[26] 52019XC1218(01). *Úřední věstník Evropské unie: Systémy elektronické identifikace oznámené podle čl. 9 odst. 1 nařízení Evropského parlamentu a Rady (EU) č. 910/2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu*. 2019. Dostupné také z: https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1577972797707&uri=CELEX:52019XC1218(01)

[27] ALLIN, Jonathan a Nick POPE. *The eIDAS Regulation For Dummies*. Chichester, West Sussex: John Wiley & Sons, 2017. ISBN 978-1-119-38087-0. Dostupné také z: http://go.thalesesecurity.com/rs/480-LWA-970/images/eIDAS-Regulation-for-Dummies-ebook.pdf

[28] TALÍŘ, Jaromír. CZ.PEPS: Základ infrastruktury pro vzájemné uznávání eID. *ISSS* [online]. [cit. 2020-01-05]. Dostupné z: https://www.isss.cz/archiv/2017/download/prezentace/cznic_talir.pdf

[29] EVROPSKÁ UNIE. *Nařízení Evropského parlamentu a Rady (EU) č. 182/2011: kterým se stanoví pravidla a obecné zásady způsobu, jakým členské státy kontrolují Komisi při výkonu prováděcích pravomocí*. In: . 2011, ročník 2011, číslo 182. Dostupné také z: https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=CELEX%3A32011R0182

[30] PRŮŠA, Jiří. CZ.PEPS: základ infrastruktury pro vzájemné uznávání eID. *CZ.NIC* [online]. [cit. 2020-01-05]. Dostupné z: https://www.nic.cz/files/nic/doc/ISSS_CZPEPS_042017.pdf

[31] PETERKA, Jiří. Český eGovernment v roce 2018: Významný milník i příchod elektronických občanek. *Lupa.cz* [online]. [cit. 2020-01-05]. Dostupné z: https://www.lupa.cz/clanky/cesky-egovernment-v-roce-2018-vyznamny-milnik-i-prichod-elektronickych-obcanek/

[32] ČESKÁ REPUBLIKA. Zákon č. 111/2009 Sb.: Zákon o základních registrech. In: *Sbírka zákonů*. 2009, ročník 2009, 33/2009, číslo 111. Dostupné také z: https://www.zakonyprolidi.cz/cs/2009-111

[33] BRECHLEROVÁ, Dagmar. *XML bezpečnost a její uplatnění v univerzitním informačním prostředí* [online]. Praha, 2008 [cit. 2020-03-25]. Dostupné z: https://insis.vse.cz/zp/portal_zp.pl?prehled=vyhledavani;podrobnosti_zp=9805;zp=9805;download_prace=1. Doktorská disertační práce. Vysoká škola ekonomická v Praze. Vedoucí práce Jiří Ivánek.

[34] Digitální podpisy. *Mendelova univerzita v Brně* [online]. [cit. 2020-04-20]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=7028

[35] BRAY, Tim a Jean PAOLI. W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Fifth Edition. 2008. Dostupné také z: https://www.w3.org/TR/xml/

[36] BRAY, Tim. INTERNET ENGINEERING TASK FORCE. *The JavaScript Object Notation (JSON) Data Interchange Format*. First edition. 2017. 2070-1721.

[37] SAFRIS, Seva. A Deep Look at JSON vs. XML, Part 1: The History of Each Standard. *Toptal* [online]. 2020 [cit. 2020-07-13]. Dostupné z: https://www.toptal.com/web/json-vs-xml-part-1

[38] SAKIMURA, Nat, John BRADLEY, Michael JONES, Breno DE MEDEIROS a Chuck MORTIMORE. OpenID Connect Core 1.0. *OpenID Foundation* [online]. 2020 [cit. 2020-07-16]. Dostupné z: https://openid.net/specs/openid-connect-core-1_0.html

[39] LODDERSTEDT, Torsten a Daniel FETT. OpenID Connect for Identity Assurance 1.0. *OpenID Foundation* [online]. 2020 [cit. 2020-03-19]. Dostupné z: https://openid.net/specs/openid-connect-4-identity-assurance-1_0-05.html

[40] HARDT, Dick. *The OAuth 2.0 Authorization Framework*. 2012. 2070-1721. Dostupné také z: https://tools.ietf.org/html/rfc6749

[41] MARTIN, James. What is IAM? Identity and access management explained. *CSO* [online]. 2020 [cit. 2020-07-20]. Dostupné z: https://www.csoonline.com/article/2120384/what-is-iam-identity-and-access-management-explained.html

[42] Server Administration Guide. *Keycloak* [online]. [cit. 2020-04-10]. Dostupné z: https://www.keycloak.org/docs/latest/server_admin/

[43] MARZOLF, Bruz. Java. *HotFrameworks* [online]. [cit. 2020-08-13]. Dostupné z: https://hotframeworks.com/languages/java

[44] OBERLE, Chris. Comparing Embedded Servlet: Containers in Spring Boot. *Baeldung* [online]. [cit. 2020-08-13]. Dostupné z: https://www.baeldung.com/spring-boot-servlet-containers

[45] 2020 Java Technology Report. *JRebel* [online]. 2020 [cit. 2020-08-13]. Dostupné z: https://www.jrebel.com/blog/2020-java-technology-report#application-server

[46] VITALE, Thomas. Keycloak Authentication Flows, SSO Protocols and Client Configuration. *Thomas Vitale* [online]. 2020 [cit. 2020-09-30]. Dostupné z: https://www.thomasvitale.com/keycloak-authentication-flow-sso-client/

[47] Keycloak authentication service. *Domino: Let your data science team use the tools they love.* [online]. 2020 [cit. 2020-09-30]. Dostupné z: https://admin.dominodatalab.com/en/4.1/keycloak.html#

[48] Jboss/keycloak. *Docker: Build and Ship any Application Anywhere* [online]. 2020 [cit. 2020-09-30]. Dostupné z: https://hub.docker.com/r/jboss/keycloak/

[49] GOOD, Michael. A Quick Guide to Using Keycloak with Spring Boot. *Baeldung* [online]. [cit. 2020-09-30]. Dostupné z: https://www.baeldung.com/spring-boot-keycloak

[50] VITALE, Thomas. Spring Security and Keycloak to Secure a Spring Boot Application - A First Look. *Thomas Vitale* [online]. 2020 [cit. 2020-09-30]. Dostupné z: https://www.thomasvitale.com/spring-security-keycloak/

[51] Securing Applications and Services Guide. *Keycloak: Open Source Identity and Access Management For Modern Applications and Services* [online]. [cit. 2020-09-30]. Dostupné z: https://www.keycloak.org/docs/latest/securing_apps/

[52] BAELDUNG. Custom User Attributes with Keycloak. *Baeldung* [online]. [cit. 2020-12-02]. Dostupné z: https://www.baeldung.com/keycloak-custom-user-attributes

Seznam obrázků

Seznam tabulek

Příloha A

Dockerfile pro Keycloak.

```
FROM registry.access.redhat.com/ubi8-minimal:8.2

ENV JDBC_POSTGRES_VERSION 42.2.5
ENV JDBC_MYSQL_VERSION 8.0.22
ENV JDBC_MARIADB_VERSION 2.5.4
ENV JDBC_MSSQL_VERSION 8.2.2.jre11

ENV LAUNCH_JBOSS_IN_BACKGROUND 1
ENV PROXY_ADDRESS_FORWARDING false
ENV JBOSS_HOME /opt/jboss/keycloak
ENV LANG en_US.UTF-8

ARG GIT_REPO=CarloS-JcU/keycloak
ARG GIT_BRANCH=master

USER root

RUN microdnf update -y && microdnf install -y glibc-langpack-
en gzip hostname java-11-openjdk-headless openssl tar which &&
microdnf clean all

ADD tools /opt/jboss/tools
RUN /opt/jboss/tools/build-keycloak.sh

USER 1000

EXPOSE 8080
EXPOSE 8443

ENTRYPOINT [ "/opt/jboss/tools/docker-entrypoint.sh" ]
```

```
CMD ["-b", "0.0.0.0"]
```

Dockerfile

Docker-compose.yml pro nasazení Keycloak.

```
version: '3.8'

volumes:
  postgres_data:
      driver: local

services:
  postgres:
      image: postgres:12.4
      volumes:
        - postgres_data:/var/lib/postgresql/data
      environment:
        POSTGRES_DB: <NÁZEV>
        POSTGRES_USER: <JMÉNO>
        POSTGRES_PASSWORD: <HESLO>

  keycloak:
      image: keycl0akn1a/idp-nia:v00
      environment:
        DB_VENDOR: POSTGRES
        DB_ADDR: postgres
        DB_DATABASE: <NÁZEV>
        DB_USER: <JMÉNO>
        DB_PASSWORD: <HESLO>
        KEYCLOAK_USER: <UŽIVATEL>
        KEYCLOAK_PASSWORD: <HESLO_UŽIVATELE>

      volumes:
        - <CESTA K FULLCHAIN.PEM>:/etc/x509/https/tls.crt
```

```
    - <CESTA K PRIVKEY.PEM>:/etc/x509/https/tls.key

  ports:
    - 80:8443
    - 443:8443
  depends_on:
    - postgres
```

docker-compose.yml

Příloha B

Dockerfile pro informační systém.

```
FROM openjdk:8-jdk-alpine
COPY <CESTA K VYTVORENEMU JARU> /app.jar
CMD ["java", "-jar", "/app.jar"]
```

Dockerfile

Docker-compose.yml pro informační systém.

```
version: '3.8'
services:
  is:
      image: keycl0akn1a/is-nia:v00
      volumes:
        - ./keystore.p12:<CESTA KE KEYSTORE.P12>
      ports:
        - 80:80
        - 443:8443
```

docker-compose.yml

Soubor *application.properties*.

```
keycloak.realm = <NAZEV REALMU>
keycloak.resource = <NAZEV KLIENTA>
keycloak.auth-server-url = <ADRESA IDP SERVERU>
keycloak.ssl-required = external
keycloak.public-client = true
keycloak.confidential-port=8443
server.port: 8443
security.require-ssl=true
server.ssl.key-store: <CESTA KE KEYSTORE>
server.ssl.key-store-password: <HESLO>
server.ssl.keyStoreType: PKCS12
server.ssl.keyAlias: tomcat
```